Notes of: Chapter 1

Software & Software Engineering

Q. What is Software?

Ans: Software is a set of programs, data and ~~isnst~~ instruction that tell a computer what to do. It makes the hardware useful.

Types of Software:

(i) System Software: This is the software that helps the computer run itself, like OS and driven

(ii) Application Software: These are the apps we used every day for work or fun, like browsers, game editors.

(iii) Engineering / scientific software: These programs sdve technical or scientific problems, like simulations, calculators, CAD tools.

(iv) Embedded Software:

Software that lives inside devices (TV, microwave, cars) and controls how they work

(v) Web/Mobile Software: Apps that run on the web or your phone, like facebook, Youtube.

(vi) AI/Data Software: Software that learns from data and makes smart decisions like chatbots, chat GPT etc.

What characteristic differentiates WebApps from other software?

Ans: WebApps run through a web browser and are delivered over a network (internet), not installed on our computer or phones.

These are the characteristic that differs a ~~sow~~ software and a WebApps.

(i) Network Independent: A WebApp lives on a network and must serve many different users over the Internet or Intranets.

(ii) Concurrency: Many users can use the WebApp at the same time.

(iii) Unpredictable Load: The number of users can jump suddenly from very low to very high.

(iv) ~~Availability~~: Performance: The WebApp must respond fast; slow pages makes user leave instantly.

(v) Immediacy: WebApps need to be built and released very quickly - sometimes within days.

P.T.O

Q3: What is Software Engineering? Is it applicable when WebApps are built? If so, how might it be modified to accommodate the unique characteristics of WebApps? [4]

[Final 20-21]

Ans:

Software Engineering: Software engineering is the disciplined way of designing, building, testing and maintaining software so that it becomes reliable ~~and~~, cost-effective, and easy to improved.

Yes, software engineering applies fully to WebApps. Because WebApps are also a software system. But software engineering needs some ~~modification~~ adjustment for WebApps because WebApps are so fast, run on browsers and serve many user at a time.

So we modify the process by:

→ Using short, itretive development cycle.

→ focusing on user experience, responsiveness and compatibility.

→ Giving more importance to security and network issus.

→ Using incremental releases because WebApps are updated continuously.

Q4. Define Product, Process, People. Explain process quality and product quality. [9] [Final 20-21]

People: These are the individuals involved in developing the software like developers, testers, designers and managers.

Process: Process is step by step method used to built the software, includes planning, designing, coding, testing and maintaining.

Product: Product is the final software system that the team delivers.

Process Quality: Process quality means how well the ~~saw~~ software development process is planned, managed and followed. A high-quality process ensure that every step like requirement gathering, design, coding, testing and ~~mafntainan~~ maintenance is done in a systematic way. In short good process quality creates a smooth workflow and building a good software.

Product Quality: Product quality describe how good the final software is from the users point of view. A high-quality product works correctly, perform fast, remains stable and easy to use. It depends on ~~ex~~ user experience. If user experience is good

then the product is good.

5. Define Software Engineering. A software product has been delivered late and exceeds the budget. Analyze possible reasons for this failure and suggest how software engineering practices could have prevented it. [5] [mid - 21-22]

Ans:

Software Engineering: check the answers of question no. 3.

The possiable reasons for failur:

1. The developer team donit understand the requirements properly.
2. Communication between developer, manager and client was very weak, causing confusion.
3. The development process was unorganized, so that ~~leading to error~~ it was leading errors and rework

4. The team have lacked of experience and knowledge about software development life cycle.

If the team follow software engineering practice they could prevent this problem. The software engineering practice teach them the following ~~thing~~ things:

1. The team should have a clear requirement analysis.
2. Need a proper communication between developers, managers and client ~~so that~~ for clear requirement and reducing misunderstanding.
3. The team should be ~~it~~ followed a systematic process to complete the development.
4. Need a proper planning, scheduling and estimate cost for completing development before ~~wh within time~~, the deadline.

General Principles of Software Engineering

Q. Write down the principle that are followed during SDLC. [Final 2021] [4]

1. The First Principle: The Reason It All Exists

→ Software should be built only if it gives real value to the user.

2. The Second Principle: KISS (Keep It Simple, stupid!)

→ Keep the design simple as possible.

3. The third Principle: Maintain the Vision

→ A project must follow a clear and consistent ~~eeh~~ architectual vision to avoid complexity.

4. 4th Principle: What You Produce, Others will Consume.

→ Design and code in a clean way so that other can read, use and maintain your code.

5. 5th Principle: ~~De~~ Be Open to the Future

→ Build the system in the flexible and extendable

6. 6th Principle: Plan Ahead for Reuse.

→ Plan so project code can be reused in future to save time.

7. 7th Principle: Think!

Always think before coding and designing.

Software Development Life Cycle (SDLC):

→ Software development Life Cycle (SDLC) is the step-by-step process used to plan, design, build, test, deploy and maintain software. It helps to build a project in a systematic and organized way.

There are six stage of SDLC:

1. Planning: In this stage, we decides why the software is needed, what are the goals

and who are the user of this project.

2. Analysis: Here we gather all the requirement from user and understand what system must do.

3. Design: In this stage, we create a ~~du~~ blueprint or ~~protot~~ prototype to see how the project will be or look like.

4. Development: This is the coding or implementation part.

5. Testing: In testing parts software is tested to find bugs, errors and performance issue

6. Deployment: The final software is delivered to client or open for ~~user.~~ ~~cu~~ customer.

~~FrameWork & Umbrella Activities:~~

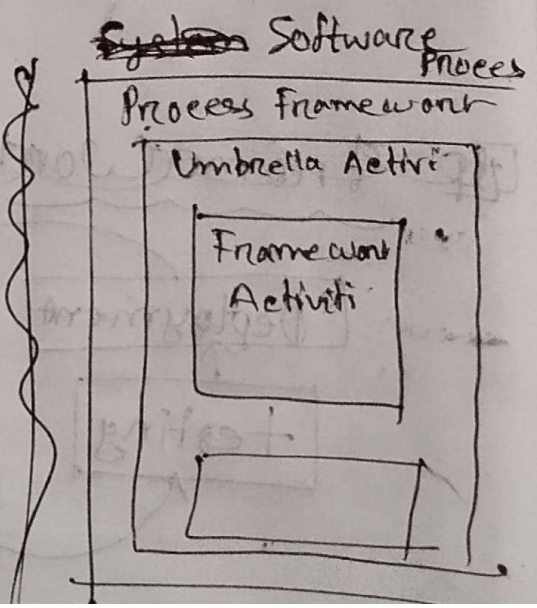
Framework & Umbrella Activities

Framework Activity: These are the core task that happen in every software development process, no matter which model we use.

There are 5 framework Activities such as 1. Communication, 2. Planning, 3. Modeling 4. Construction, & 5. Deployment.

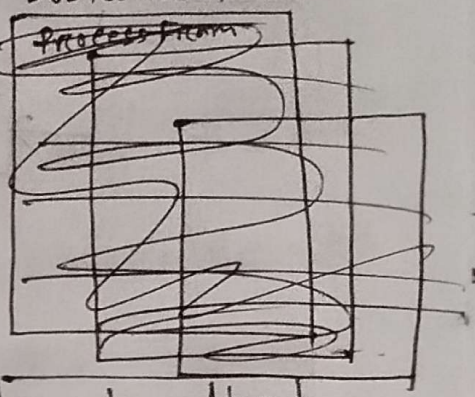
Umbralla Activity: Umbrella activities are supporting task that run across all phases of SDLC. There are 8 Umbrella activities. Such as,

1. Project Tracking & control
2. Risk management
3. Quality Assurance
4. Technical Reviews.
5. Measurement



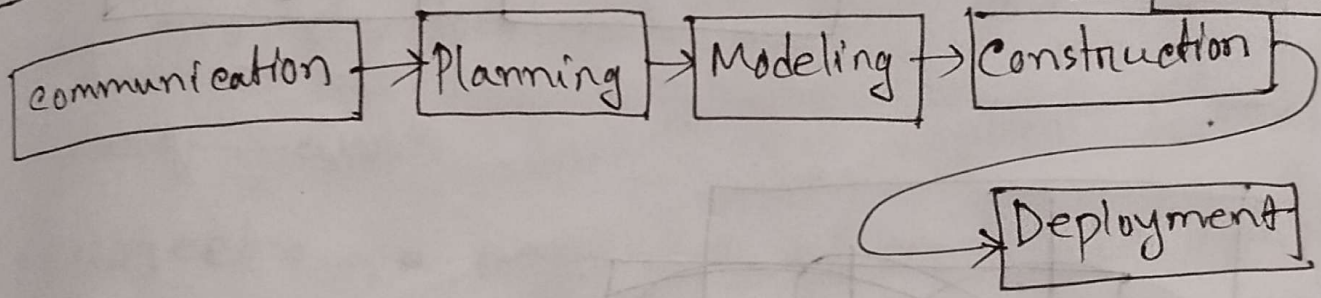
6. Reusability Management
7. Configuration Management
8. ~~Cutto~~ Work Product Preparation.

Software Process

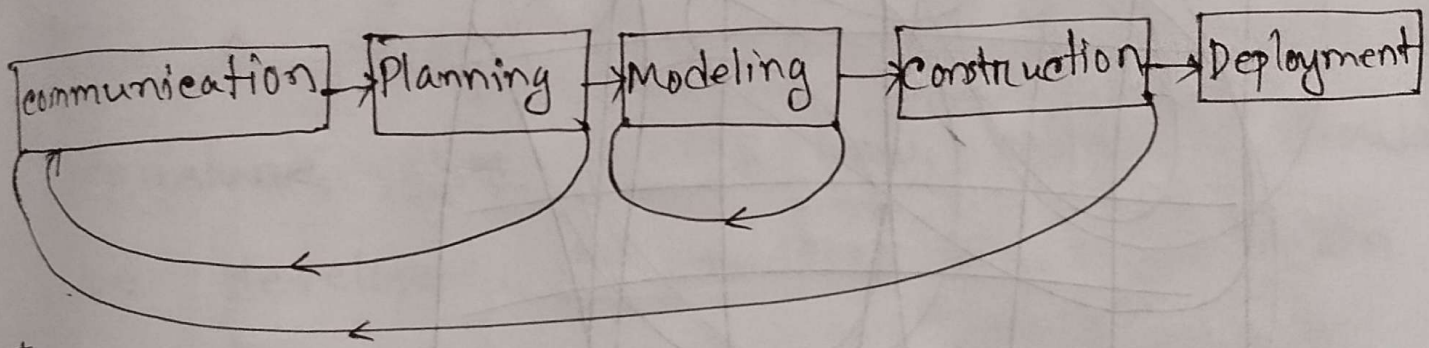


(d) Process Flow:

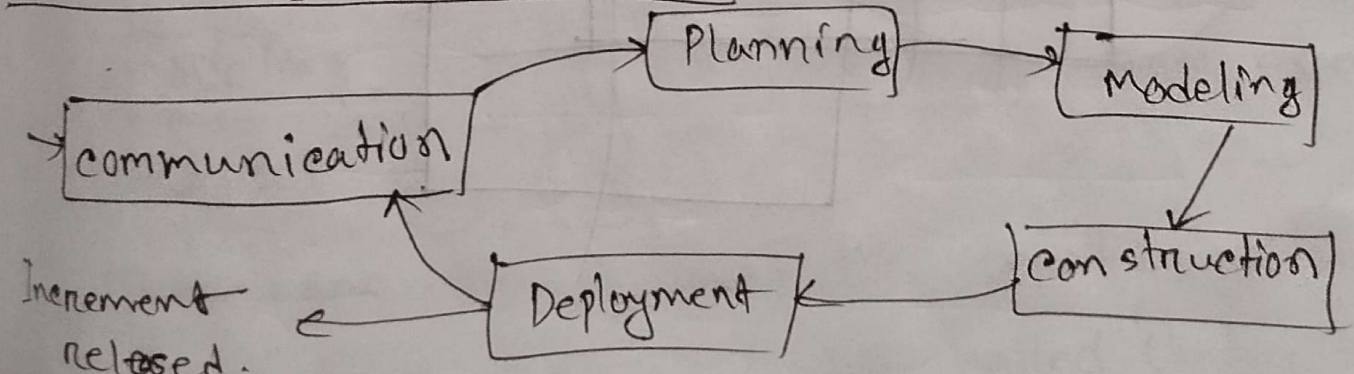
Linear Process Flow:



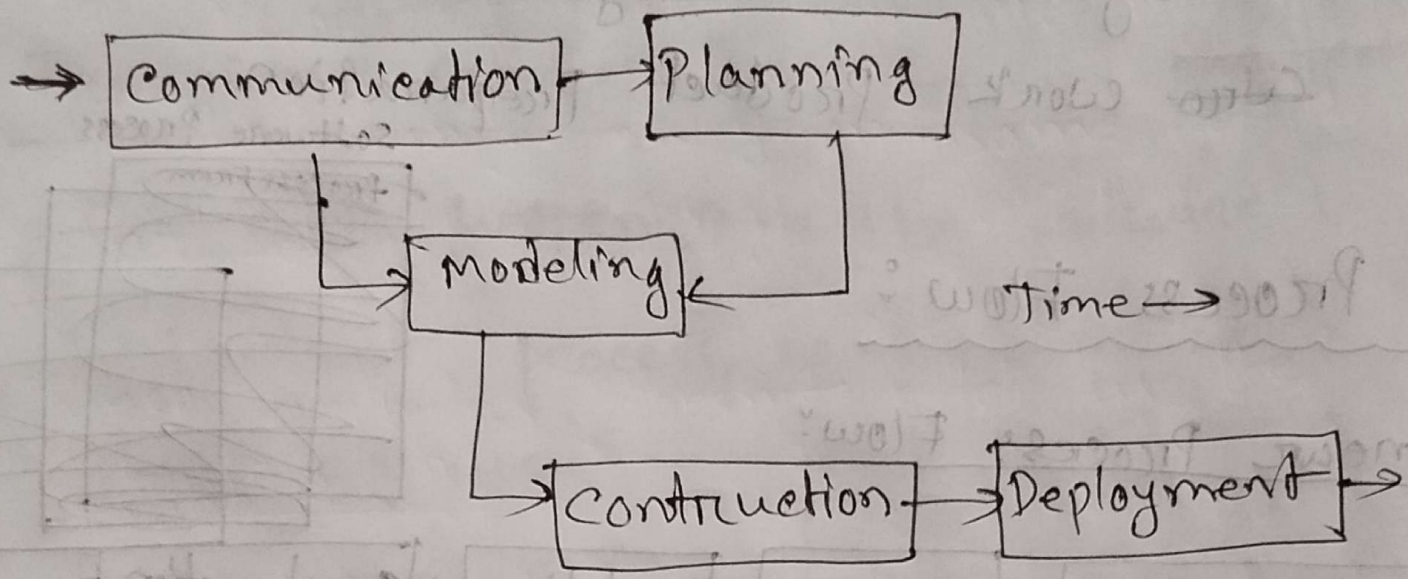
~~Ite~~ Iterative Process Flow:

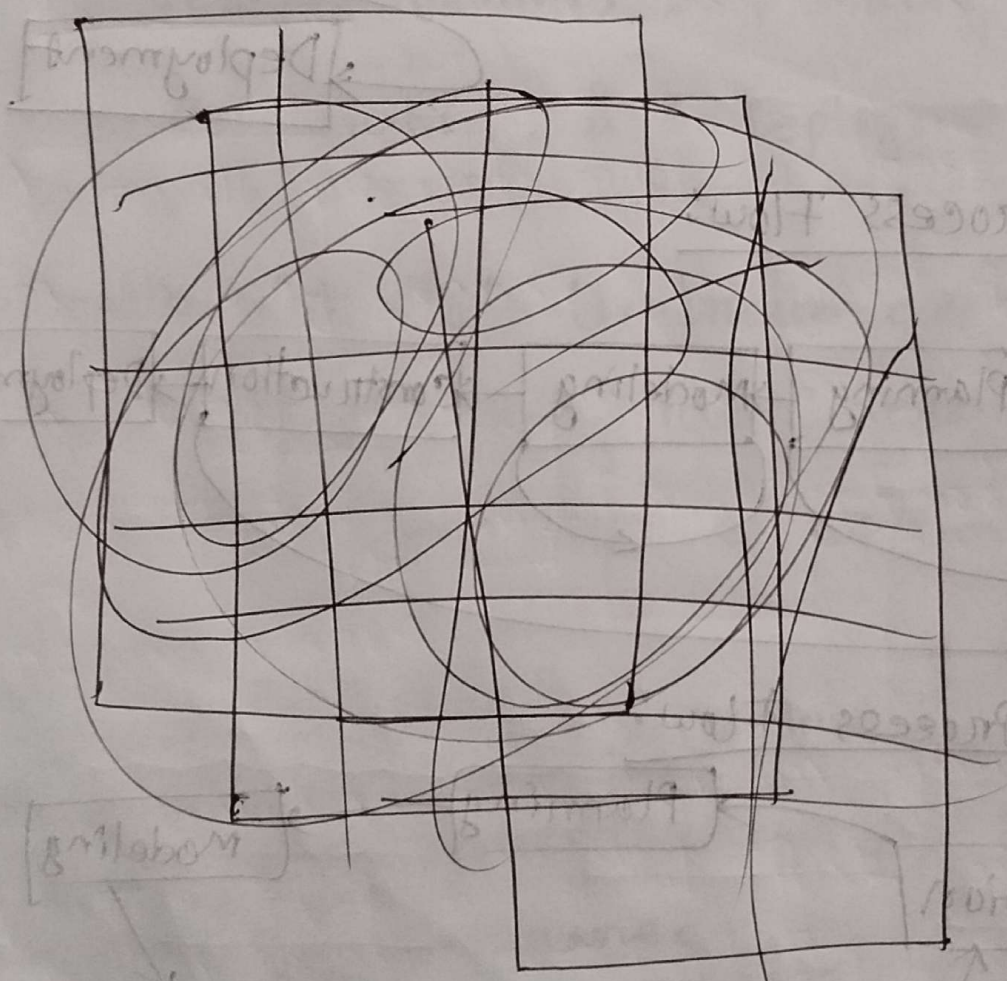


Evolutionary Process Flow:



Pallalel process Flow:





[4] Process Model:

1. The Waterfall Model:

Q: 7: Describe a process framework in your own words. When we say that framework activities are applicable to all project, does this mean that the same work tasks are applied for all projects, regardless of size and complexity? [Mid 21-22] [5]

Ans: A process framework is a basic structure that guides how software should be developed. It provides a set of main activities like communication, planning, modeling, coding and testing - that every project should follow.

These are the framework activities. Also there is a supporting task called Umbrella Activity.

Yes, these framework activities are common to all project but they are not applied in the same way on same depth for every project.

These activities doesn't tell us the exact steps but gives a general roadmap ~~to~~ so that ~~the~~ the team can understand ~~do know~~ where is the starting point.

A small project may do these activities in a very simple way and quickly, but in large and complex project needs more deep planning, ~~and~~ time, effort and documentation.

So, activities are remain same for all but level of detail, risk and complexity are vary from project to project.

Process Model:

1. The Waterfall Model:

Note

The waterfall Model is a step-by-step linear process that start with collecting requirement and moves through planning, modeling, construction and deployment. It is the oldest development model and it process ~~are~~ is rigid and best for when the requirements are clear.

Mid 20-21 Q. no. 3

Ans:

Phases of Waterfall Model is given below:

1. Communication: This phase is the project initiation where requirements are gathering by communiting with client, ~~manager~~ or stake holders.

2. Planning: In this phase, analyze the requirement estimating the cost, ~~sh~~ scheduling and make tracking of workflow.

3. Modeling: ~~Analyze~~ Analyzing which model and design will be implement for this project.

4. Construction: This is the coding phase. Also the testing occurs in this phase in waterfall model.

5. Deployment: In this phase, project~~s~~ ~~are~~ is handovered to the client.

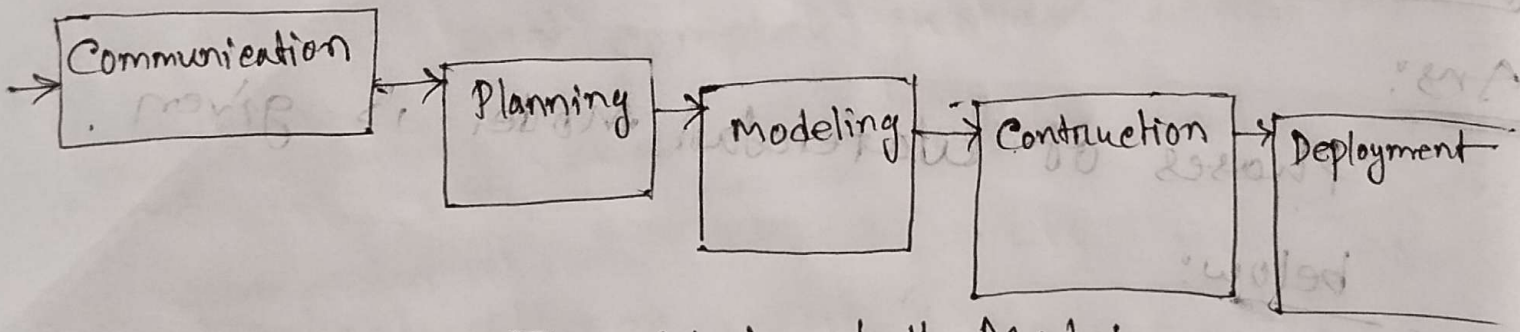


Fig: Waterfall Model.

As the ~~a~~ requirements are clear and stable ~~of~~ this banking ~~a~~ project, the Waterfall model is very suitable for this type of project.

Because Waterfall model is suitable when requirements are clean ~~and~~, stable and not changable

In waterfall model, every steps starts ~~be~~ when the previous steps completely done. So there is no flexibility to change something later.

~~For this reason, Waterfall model is to suitable~~

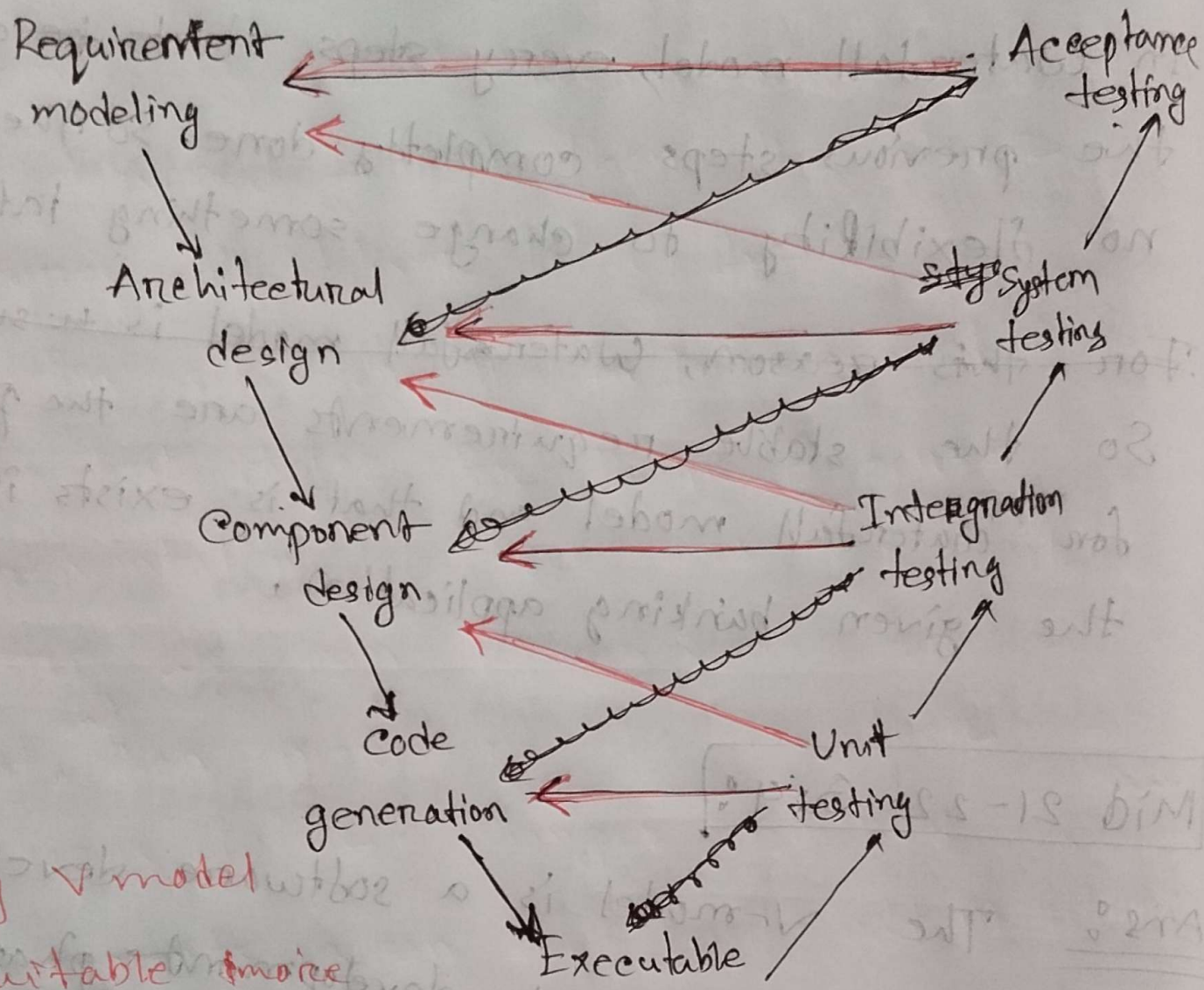
So the stable requirements are the precondition for waterfall model and that is exists in the given banking application.

Mid 21-22 Q:4°.

Ans: The V-model is a software development process where each development phase on the left side has a matching testing phase on the right side.

It shows a "V" shape where, requirements, architecture, component and code are in the left side and Acceptance, system, integration and unit testing on the right side.

Diagram: Note



Why V model is suitable more than waterfall model in medical sector?

Ans: Medical software must be safe and 100% error free and the 'V' model gives a very strong validation at each process. In 'V' model testing for each phase are not delayed like waterfall model. So,

it ~~makes~~ the software system's accuracy and reliability.

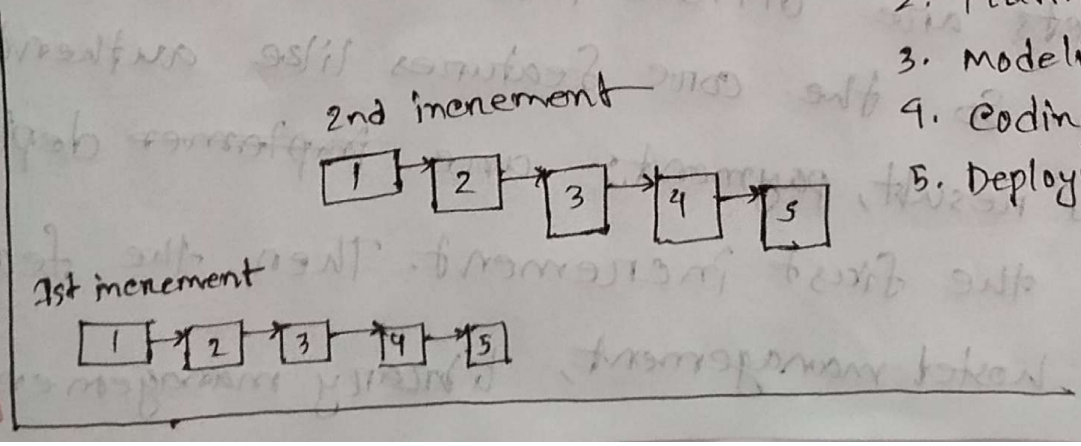
This system ensures strict validation, better risk management than Waterfall model. ~~due to~~ In 'V' model bugs and errors are caught early, makes the system medical standard and ensure patient safety.

Mid 21-22 Q:5

Ans: For a University Management System where some core features must be delivered quickly and extra features can be added later, the Incremental Model is the best ~~co~~ choice for this project.

1. Communication
2. Planning
3. Modeling
4. Coding
5. Deployment

Diagram of Incremental Model



Why it fits?

As client doesnt wait for the entier system, they want some core features are delivered first then the others features will deliver later. In this situation, the Incremental Model is the best choice for them. Because incremental model gives us this flexibility where we can deploy some core first and then increments the other features later. Changes and new requirements can easily add in future.

How the model manage the project?

In incremental modeling process entier projects are divides into small and managable parts. So the core features like authentication, view result, payments are deployed into the first increment. Then the features like hostel management, library management

functionalities are added later. &

During every increment developers can get feedback and can add new requirement based on the user needs.

So, risks are lower, ~~testing and~~ coding, testing deployment and new requirement gathering can possible ~~pat~~ parallelly.

[4] Spiral Model:

1. What is spiral model?

The **spiral model** is a software development process that works in repeated cycles. Each cycle includes planning, risk analysis, development, and evaluation.

2. Where is Spiral Model Used?

Spiral model is used in projects that are:

- ✓ **Large and expensive**
- ✓ **High-risk**
- ✓ **Complex**
- ✓ **Not fully understood at the beginning**
- ✓ **Require frequent changes**
- ✓ **Need prototypes**

Examples:

- Banking systems
- Defense projects
- Missile control systems
- Large enterprise apps
- Projects with unknown technology

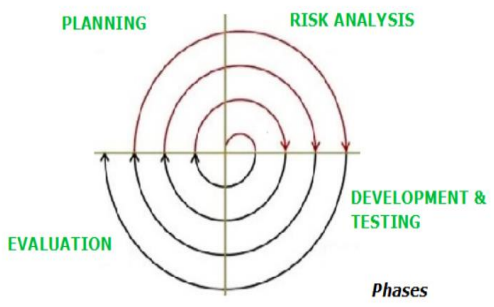
3. Key Points to Identify When Spiral Model Is Suitable

You can easily tell the spiral model is suitable when:

- ➢ **Requirements are unclear or likely to change**
- ➢ **Project has high risk (technical, financial, or safety)**
- ➢ **Prototyping is needed to understand the solution**
- ➢ **Large-scale project with many modules**

4. How the process of the spiral model works? / Explain the every phases of Spiral Model.

Planning: The planning phase includes collecting requirements from the client and checking feasibility. The team then estimates the cost, time, and resources needed.



Risk Analysis: The risk analysis phase identifies possible risks in the project and prepares strategies to handle or reduce those risks.

Development & Test: The development and testing phase builds the planned part of the software and tests it to find defects.

Evaluation: The evaluation phase shows the software or prototype to the customer for feedback. This feedback helps decide what changes or additions are needed in the next cycle.

5. Write a simple scenario. Then answer this question based on the scenario: Why the spiral model is suitable for this scenario?

A company wants to build a **secure online banking system** with **AI-based fraud detection**, but many requirements are **unclear** and the security **risks are high**.

Why the Spiral Model Is Suitable for This Scenario:

The project is suitable because:

- ✓ Requirements (security rules, AI features) may change over time.
- ✓ It has **high risk**, especially in security and financial transactions.
- ✓ Prototypes are needed to validate user interfaces and AI models.
- ✓ Continuous customer and expert feedback is required.
- ✓ The cost of failure is high, so **risk analysis in every cycle** is essential.

6. Difference Between Incremental Model and Spiral Model.

Aspect	Incremental Model	Spiral Model
Development Approach	Build the system in **small parts** (increments).	Build in **repeated cycles** with planning + evaluation.
Risk Handling	**Very little risk analysis**.	Strong focus on **risk analysis in every cycle.**
Project Size & Complexity	Best for **small/medium** projects.	Best for **large, complex, high-risk** projects.
Prototypes	Not always needed.	Used often to reduce risks.
Focus	**Fast delivery** of working modules.	**Slow delivery** due to repeated cycles
Mostly Used In	Social Media apps, gaming apps etc	Defense, banking system, AI

---

7. What is agile model?

The Agile model is a software development approach where work is done in small, quick cycles, and the team delivers a working part of the software in each cycle.

8. When the agile model is suitable?

Agile is suitable when requirements change frequently, when the customer needs regular updates, when fast delivery is necessary, and when the project needs flexibility instead of strict planning.

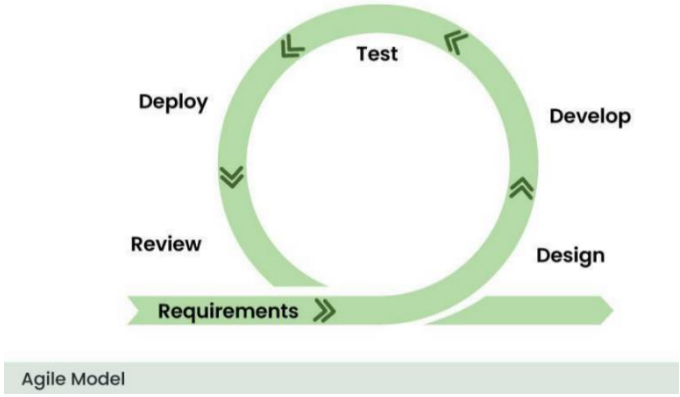
9. What are those key points that I can easily identify a scenario based on agile model?

- ✓ Requirements are not fixed and may changes **rapidly**.
- ✓ The customer wants to see the software **quickly and give feedback** often.
- ✓ The team prefers collaboration**, daily communication**, and **quick decisions**.

10. Write down the procedure / process of agile model?

Steps of the Agile Model (Simple Explanation)

- **Requirements:** In this step, the team discusses with the customer to collect the basic requirements but the goal is not to gather everything at once.
- **Design:** The team creates a simple and flexible design for the features planned in that iteration.
- **Develop: Developers write the code for the selected features**. The work is done in small parts, so the software grows step by step with every iteration.
- **Test: The developed features are tested immediately.** The team checks for errors, fixes bugs, and makes sure the feature works correctly before moving forward.
- **Deploy:** After testing, the working feature is deployed or delivered to the customer.
- **Review: The team reviews the completed iteration with the customer**. Feedback is collected, changes are noted, and new requirements are identified.



Agile Model

11. Imagine you are leading a software development team for a project with rapidly changing requirements. Which software process model would you recommended and why? Provided detailed explanation of how this model accommodate changes efficiently an ensure successful project delivery.

In this scenario, the **Agile model** is most suitable because the project has **rapidly changing** requirements. Agile works in small iterations, allowing the team to receive new requirements, update the product, and deliver improved versions quickly.

Agile handles changes efficiently because each iteration includes planning, designing, coding, and testing. If requirements change, the team adjusts in the next iteration without delaying the project. Frequent customer feedback reduces rework, improves quality, and ensures the project stays on track. Agile's flexibility, fast updates, and close collaboration make it ideal for successful project delivery.

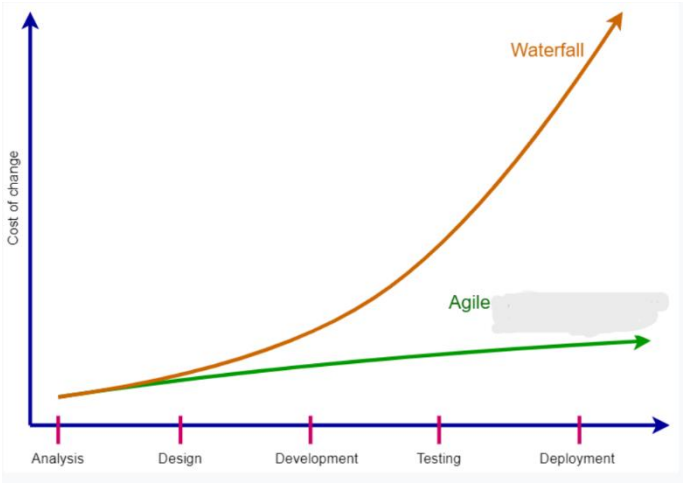
What is Agility and What is the Cost of Change with Agile model?

Concept of Agility

Agility means the ability to adapt quickly when requirements change. In Agile development, the team works in short cycles, takes regular feedback from the customer, and updates the product step by step. This makes the software flexible and easy to improve throughout the project.

Agile and Cost of Change

In traditional models (like Waterfall), the cost of change becomes very high if requirements change at the end of the project. But in Agile, the cost of change stays low even in later stages. This is because development happens in small iterations, and changes can be applied immediately in the next cycle. Continuous testing and feedback also prevent big mistakes, reducing the cost of fixing problems.



Write down the Principles of Agile Model.

Here are the 12 Agile Principles, each into one simple, easy line:

1. Deliver useful software **early** and continuously to keep the **customer satisfied**.
2. Accept **changes anytime** because they help improve the product.
3. Release working software in **short cycles**.
4. Ensure business people and developers **work together daily**.
5. Build **teams with motivated people** and give them the support they need.
6. Use **face-to-face communication** for fast and clear information sharing.
7. **Measure progress** mainly by delivering working software.
8. Maintain **a steady and manageable work pace** throughout the project.
9. **Focus on good design** and technical quality to **stay adaptable**.
10. **Keep things simple** by avoiding unnecessary work.
11. Allow teams **to organize themselves** to produce the best results.
12. **Regularly review the process** and improve how the team works.

What is the XP Process?

The XP (Extreme Programming) process is an Agile software development approach that focuses on fast delivery, frequent customer feedback, simple design, continuous testing, and close teamwork. It works in short cycles so the team can quickly adapt to changes and deliver high-quality software step by step.

Write down the XP Processes.

XP (Extreme Programming) Process – Simple Explanation

1. Planning: The customer explains what features they want, and the team breaks them into small user stories so that they can complete it quickly.

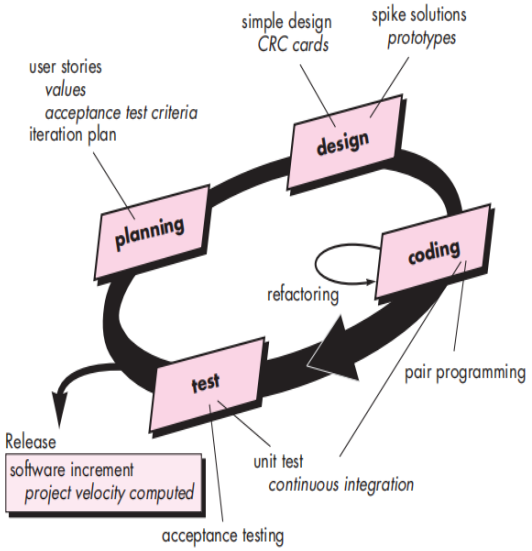
2. Design: The team designs only what is needed for the current feature.

3. Coding: Developers write code in pairs (pair programming), use coding standards and avoid errors.

4. Testing: Tests are written before coding (test-driven development), and the system is tested continuously.

5. Release: Small updates are delivered often so the customer can check progress early.

FIGURE 3.2
The Extreme Programming process



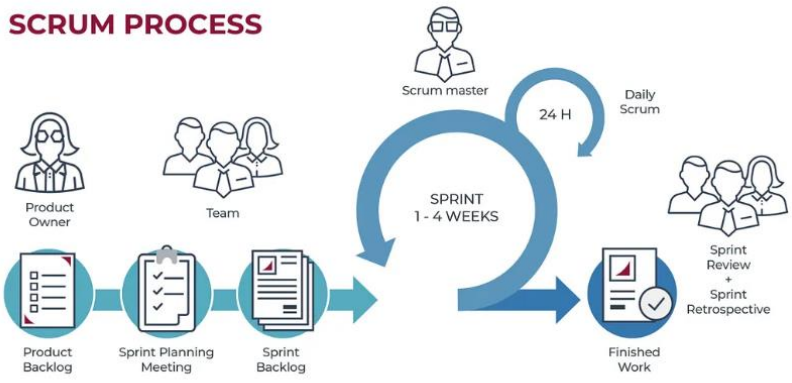
6. Feedback & Improvement: The team takes customer feedback and improves the next iteration.

What is Scrum?

Scrum is an Agile process model used to develop software in small, manageable pieces called sprints (2–4 weeks). It focuses on teamwork, quick updates, continuous improvement, and frequent delivery of working software.

Scrum Procedure (Simple Steps)

1. **Product Backlog:** The customer lists all features they want in the system.
2. **Sprint Planning:** The team chooses a small set of features from the backlog to complete in the next sprint.
3. **Sprint (Development Cycle):** The team works for 1–4 weeks to design, develop, and test the chosen features.



4. **Daily Scrum Meeting:** A short daily meeting (10–15 minutes) where team members share progress and plan the day.
5. **Sprint Review:** The team shows the completed work to the customer and gets feedback.
6. **Sprint Retrospective:** The team discusses what went well and what needs improvement before starting the next sprint.

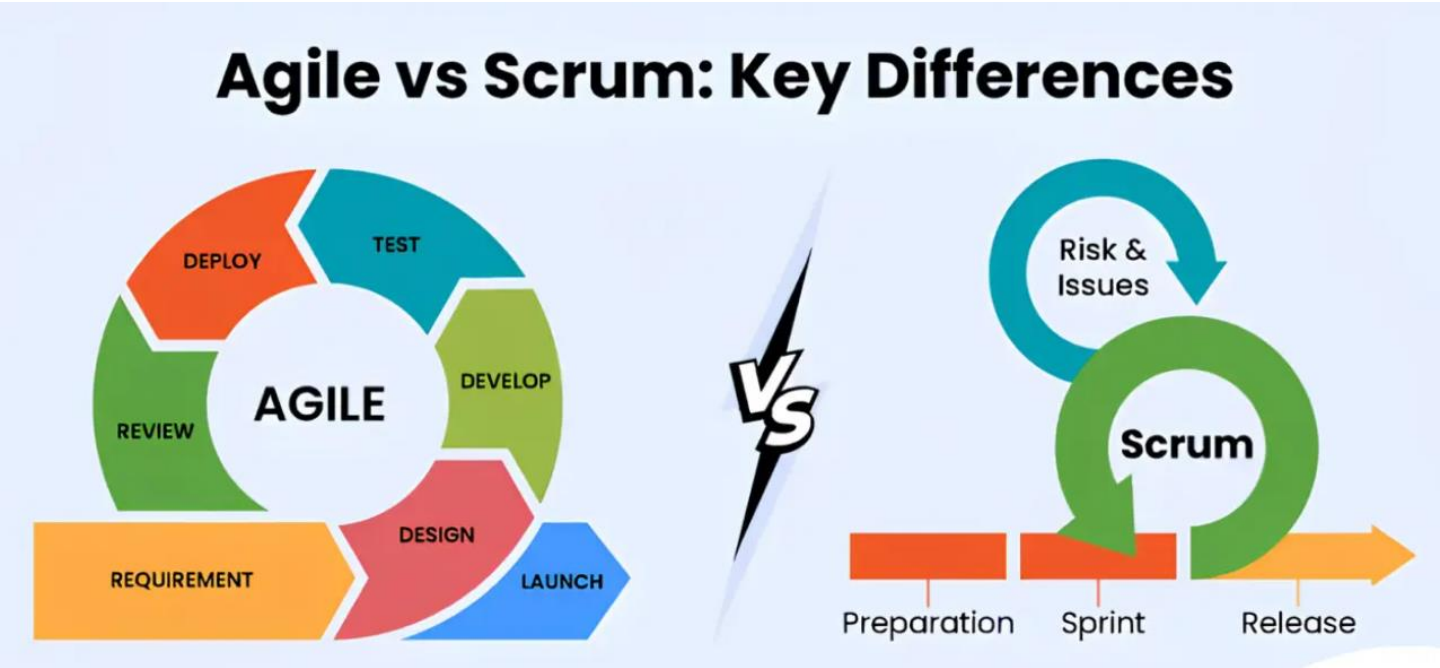
Scenario (When Scrum Is Suitable)

Imagine you are developing an online learning platform where the client keeps changing or adding new features—like quizzes, dashboards, live classes, notifications, etc. The requirements are not fixed and will grow over time.

Why Scrum is suitable here:

- ✓ The project **needs frequent updates** because new ideas come regularly.
- ✓ The client can see **progress every sprint** and change priorities.
- ✓ The team can **adapt quickly** if the client wants to modify or add features.
- ✓ **Continuous feedback improves** the product step by step.

Because Scrum supports flexibility, fast delivery, and constant customer involvement, it fits perfectly for this kind of evolving project.

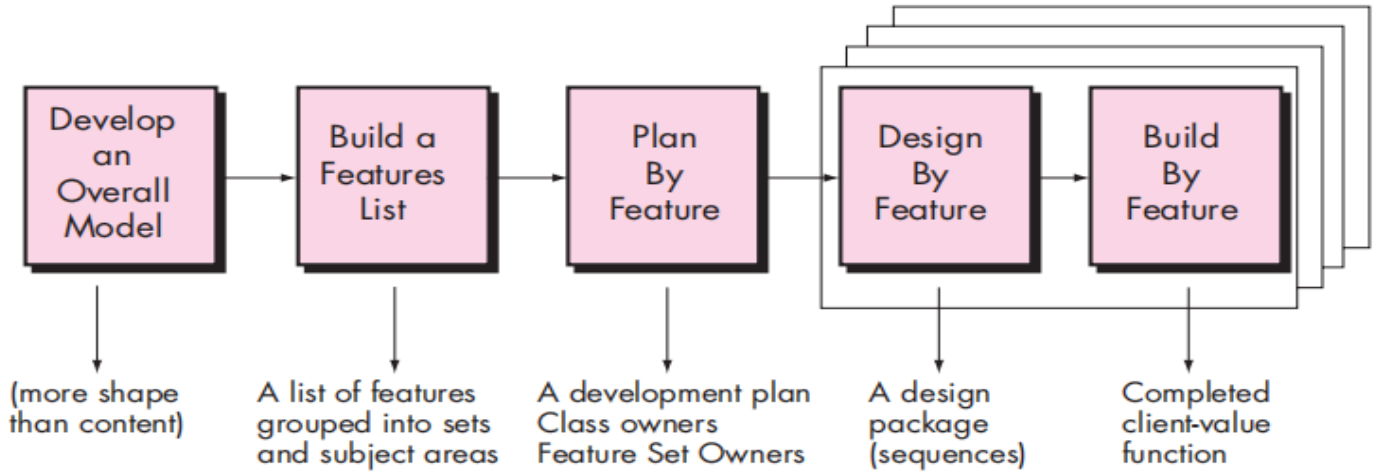


What is Feature Driven Development (FDD)?

Feature Driven Development (FDD) is an Agile method that builds software feature by feature in short, repeated cycles. FDD focuses on planning, designing, and building these small features one at a time to ensure fast progress and high quality.

Steps of the FDD Process (Simple)

1. **Develop an Overall Model:** Understand the system and create a simple high-level design.
2. **Build the Features List:** Break the system into small user-focused features.
3. **Plan by Features:** Decide the order of features and assign developers.
4. **Design by Feature:** For each feature, create a small design before coding.
5. **Build by Feature:** Developers implement, test, and deliver each feature.



When FDD is Suitable (Scenario)

Imagine your team is developing a shopping app with many small functions such as viewing products, adding to cart, checking out, tracking orders, etc. These functions can be broken into clear features, and each feature can be built separately.

Why FDD fits this scenario:

- ✓ The system has many small, well-defined features.
- ✓ The project needs quick and continuous progress.
- ✓ Each feature can be designed, developed, and tested independently.
- ✓ The customer can see completed features regularly.

What do you mean by Risk Analysis and Management? What steps are involved in it?

Risk analysis and management means identifying possible problems that could harm the project (like delays, technical issues, cost increase), and preparing ways to reduce or avoid them. The goal is to protect the project from failures.

Steps involved:

1. **Risk Identification:** Find out all possible risks, unclear requirements, schedule problems, etc.
2. **Risk Analysis**: Study each risk to understand how it happens and how much damage it may cause.
3. **Risk Prioritization:** Decide which risks are most serious and need quick attention.
4. **Risk Planning:** Prepare strategies to avoid the risk or reduce its impact.
5. **Risk Monitoring:** Continuously check if the risks are increasing or decreasing, and update the plan.

Define Project Management. Write down the good qualities of a project manager.

Project management is the process of planning, organizing, tracking, and controlling all activities of a software project so that it finishes on time, within budget, and meets customer requirements.

Qualities of a Good Project Manager:

- ✓ Good communication skills.
- ✓ Strong leadership and decision-making ability.
- ✓ Ability to plan and organize work properly.
- ✓ Good problem-solving skills.
- ✓ Able to handle risks and unexpected issues.
- ✓ Motivates the team and keeps them productive.
- ✓ Understands both technical and business needs.

Why software project planning and tracking is necessary in software engineering?

Project planning is important in software engineering because it defines what work needs to be done, the order of tasks, required resources, time, budget, and responsibilities. A clear plan ensures everyone knows their role and project goals. It also helps the team anticipate problems, use resources efficiently, and set realistic deadlines. Breaking the project into small tasks makes it easier to track progress and stay focused.

Tracking the project ensures work is progressing as planned. It helps detect delays, unexpected issues, or extra costs early so corrections can be made. Tracking also keeps stakeholders informed about progress and risks. Without proper planning and tracking, projects may face delays, cost overruns, poor quality, or fail to meet customer expectations. Together, planning and tracking increase the chances of delivering software successfully and on time.

What is RMMM and the primary objectives of RMMM plan?

RMMM stands for **Risk Mitigation, Monitoring**, and **Management**. It is a structured approach in software project management **used to handle risks effectively**. The RMMM plan can either be part of the overall software project plan or a separate document. It records all activities related to risk analysis and provides guidance for managing risks throughout the project lifecycle.

Primary Objectives of the RMMM Plan:

1. **Risk Mitigation (Problem Avoidance):** Define and implement strategies to reduce the likelihood or impact of identified risks before they occur.
2. **Risk Monitoring (Project Tracking):**

   Track risks during the project to ensure:

   - Predicted risks actually occur or are avoided.
   - Risk aversion or mitigation steps are correctly applied.
   - Information is collected to improve future risk analysis.

3. **Risk Management (Problem Allocation):** Identify which risks caused which problems during the project and use this information to guide corrective actions.