DOMS | Page No.
Date / /

Software Engineering - QB.

Q2) Explain with a diagram spiral process model in detail & the kind of applications they are most suitable for.

Ans:
- Spiral model is a combination of iterative & waterfall model
- It provides potential for rapidly development of increasingly more complete version of the software
- Using spiral, software developed in as series of evolutionary release
- early iteration, release might be on paper/ prototype
- later iteration, more complete version of software
- Divided into framework activities (C, P, M, C, D). Each activity represents one segment
- Evolutionary process begins in a clockwise direction, beginning at the center risk.
- First circuit around the spiral might result in development of product specification.
- Subsequently, develop a prototype and then progressively more sophisticated version of software.
- Unlike other process models that end when software is delivered.
- It can be adapted to apply throughout the life of software.

Concept Development Project:

- Start at the core and continues for multiple iterations until it is complete.
- If concept is developed into an actual product, the process proceeds outward on the spiral.

New Product Development Project:

- New product will evolve through a no. of iterations around the spiral
- later, a circuit around spiral might be used to represent "Product Enhancement Project"

Product Enhancement Project:

- There are times when the process is dormant or software team not developing new things but change is initiated, process starts at appropriate entry point.
- Spiral models use prototyping as a risk reduction mechanism but, more important, enables the developers to apply the prototyping approach at each stage in the evolution of the product.

- It maintains the systematic stepwise approach suggested by the classic life cycle but also incorporates it into an iterative framework activity

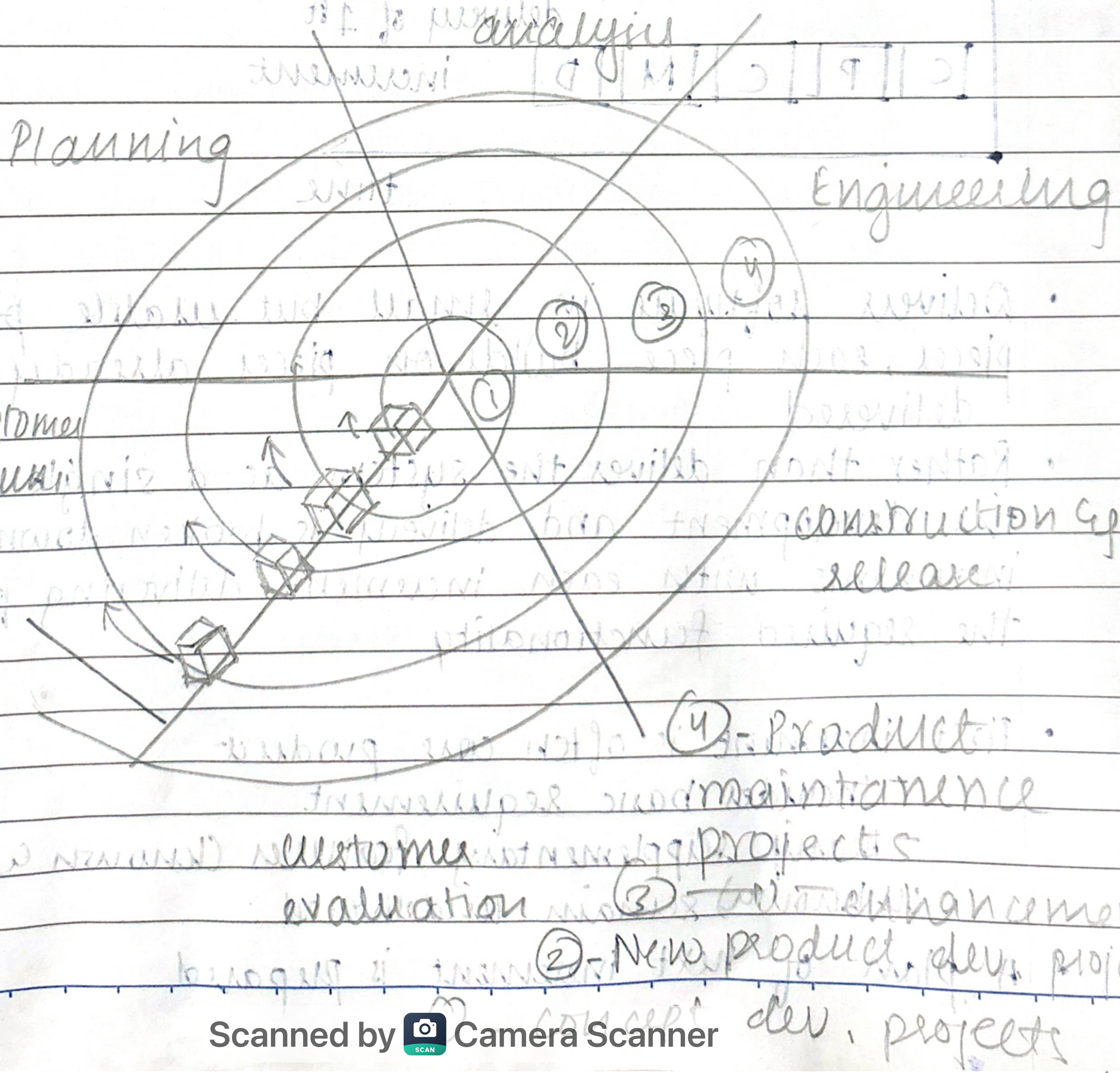
- If risks cannot be resolved, project is immediately terminated.

→ Advantages

- more flexible to changing requirements
- risk management is easies

→ Disadvantages

- It is difficult to manage development process
- Spiral can run indefinitely.

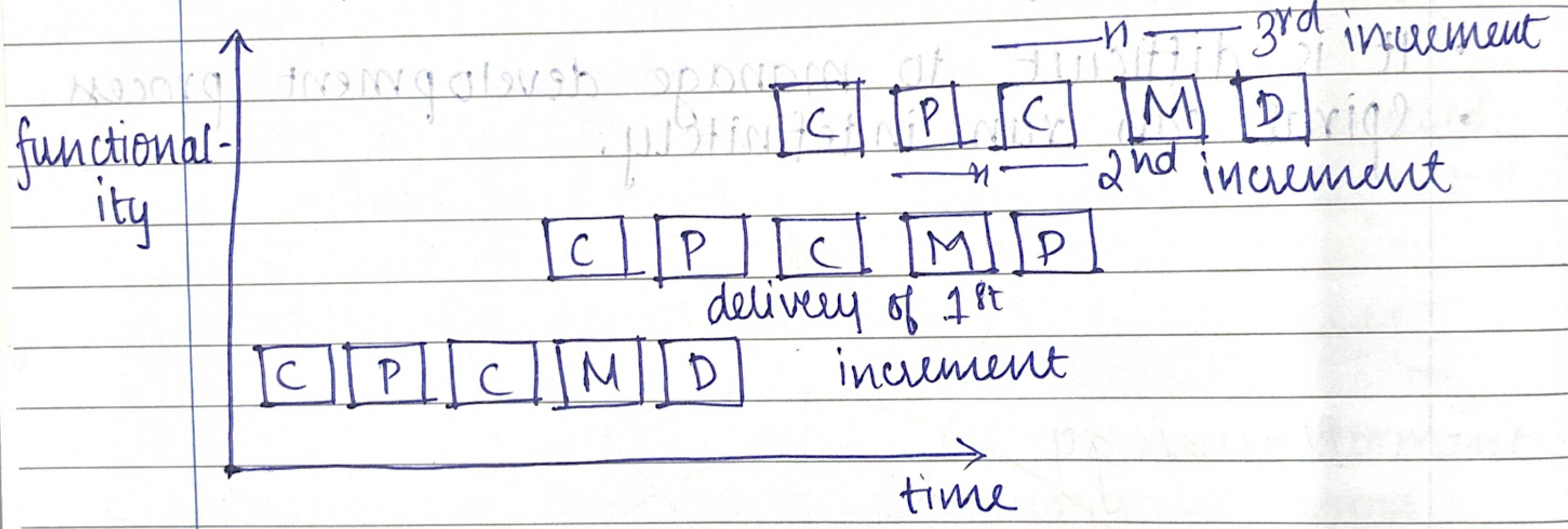


Q3)	Waterfall	Spiral
Dev. & Style:	Linear & Sequential	Iterative with risk analysis and feedback loops
Flexibility:	Inflexible - hard to go back to prev. phase	Highly flexible - allows revisiting and refining each phase
Risk Manag.:	Minimal	Core focus.
Testing:	Done after implementation is complete	Done in every cycle
Cost & Time Est:	Easier (if req. are stable)	More complex due to iteration & risk assessment.
Prototyping:	No prototyping; all decisions made early	Encourages early prototyping in each cycle.

DOMS | Page No. 4
Date 21/7/25

Q4. Discuss Incremental Process Model for Software development with merits & demerits.

Ans: C - Communication
P - Planning
M - Modelling
C - Construction
D - Deployment



- Delivers software in small but usable pieces, each piece builds on pieces already delivered
- Rather than delives the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality

- First increment is often core product
  - Includes basic requirement
  - many supplementary features (known & unknown) remain undelivered.
- A plan of next increment is prepared

- Modifications of the first increment
- Additional features of the first increment
- It is particularly useful when enough staffing is not available for the whole project
- Increment can be planned to manage technical risks
- Incremental model focuses more on delivery of operation product with each increment.
- User requirements are prioritised and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve
- Customer value can be delivered with each increment so system functionality is available earlier
- Early increments act as a prototype to help elicit requirements for later increments
- Lower risk of overall project failure
- The highest priority system tend to receive the most testing.

Merits -

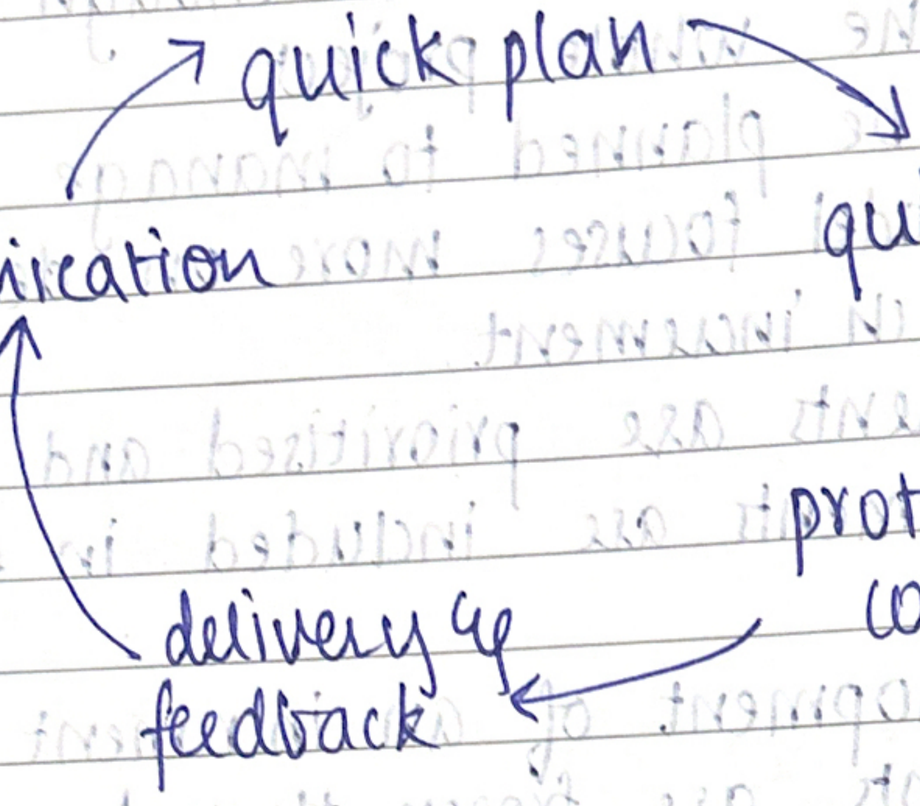
1. Early partial product Delivery
2. Easier testing & debugging

Demerits -

1. Requires good planning
2. Integration issues.

DOMS | Page No. 6
Date / /

Q5. Discuss prototype model with merits & demerits.



- Best approach when-

→ Objectives defined by customer are general but does not have details like input, processing or output requirement
→ Developer may be unsure of the efficiency of an algorithm, OS, or the form that human machine interaction should take

- Can be used as standalone process model.
- This model assists s.e & customer to better understand what is to be built when requirements are fuzzy.
- Prototyping starts with communication, b/w a customer and software engineer to define overall objectify, identify requirements and make a boundary
- Going ahead, planned quickly and modeling occurs
- Quick design leads to prototype construction

- Prototype is deployed and evaluated by the customer/user
- Feedback from customer/end user will refine requirement and that is how iteration occurs during prototype to satisfy the needs of the user
- Prototype can be served as "the first system"
- Both customer & developers like the prototyping paradigm
  → customer/end user gets a feel for the actual sys.
  → developer get to build something immediately.

DeMerits -

1. Incomplete requirements understanding
   - Frequent changes and feedback can lead to uncontrolled growth in project scope, delaying project delivery.
   - Customers may only focus on what they see in the prototype and may not consider overall system requirements

Merit -

1. Better Requirement Understanding
   - Helps clarify user requirements when they are vague/not well understood at the beginning
2. Early feedback from users
   - Users can interact with a working prototype early, which helps gather useful ~~information~~ feedback and reduce misunderstandings.

Q6: Explain waterfall model & its phases in detail.

→ Requirement Analysis and Definition:
The system services, constraints are defined by customers with system users

→ Scheduling tracking:
- Require action to maintain schedule
- Assessing progress against the project plan

→ System and Software Design:
- It establishes overall system architecture
- Software design involves fundamental system abstractions and their relationships.

→ Integration and System testing:
- The individual program unit/programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

→ Operation & Maintenance:
- Normally this is the longest phase of the software life cycle.
- The system is installed and put into pratical use. Maintanence involves correcting errors which were not discovered in the earlier stages of the life-cycle.

communication → project initiation requirement gathering
planning → estimating scheduling tracking
modeling → analysis design
construction → code test
deployment → delivery support feedback

- The nature of the requirements will not change very much during development
- The model implies that you should attempt to complete a given stage before moving on to the next stage
  → does not account for the fact that requirements constantly change
  → It also means that the customers cannot use anything until the entire system is complete
- The model implies that once the product is finished, everything else is maintanence.
- Surprises at the end are very expensive
- Some teams sit ideal for other teams to finish

∴ This model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.

Demerits:

1. Rigid and Inflexible → Difficult to go back to a previous phase once its completed.
2. Late Testing → Testing is done at the end so issues may be discovered later in the project

Merits:

1. Simple and easy to understand → The step-by-step process is straightforward making it easy to
2. Easy to manage manage.

Bcoz of its rigid structure, the model is easy to schedule, ma

Q1) Develop a SRS.

—— X Doc given of IEEE format X ——

Q7) (A) Agile process:

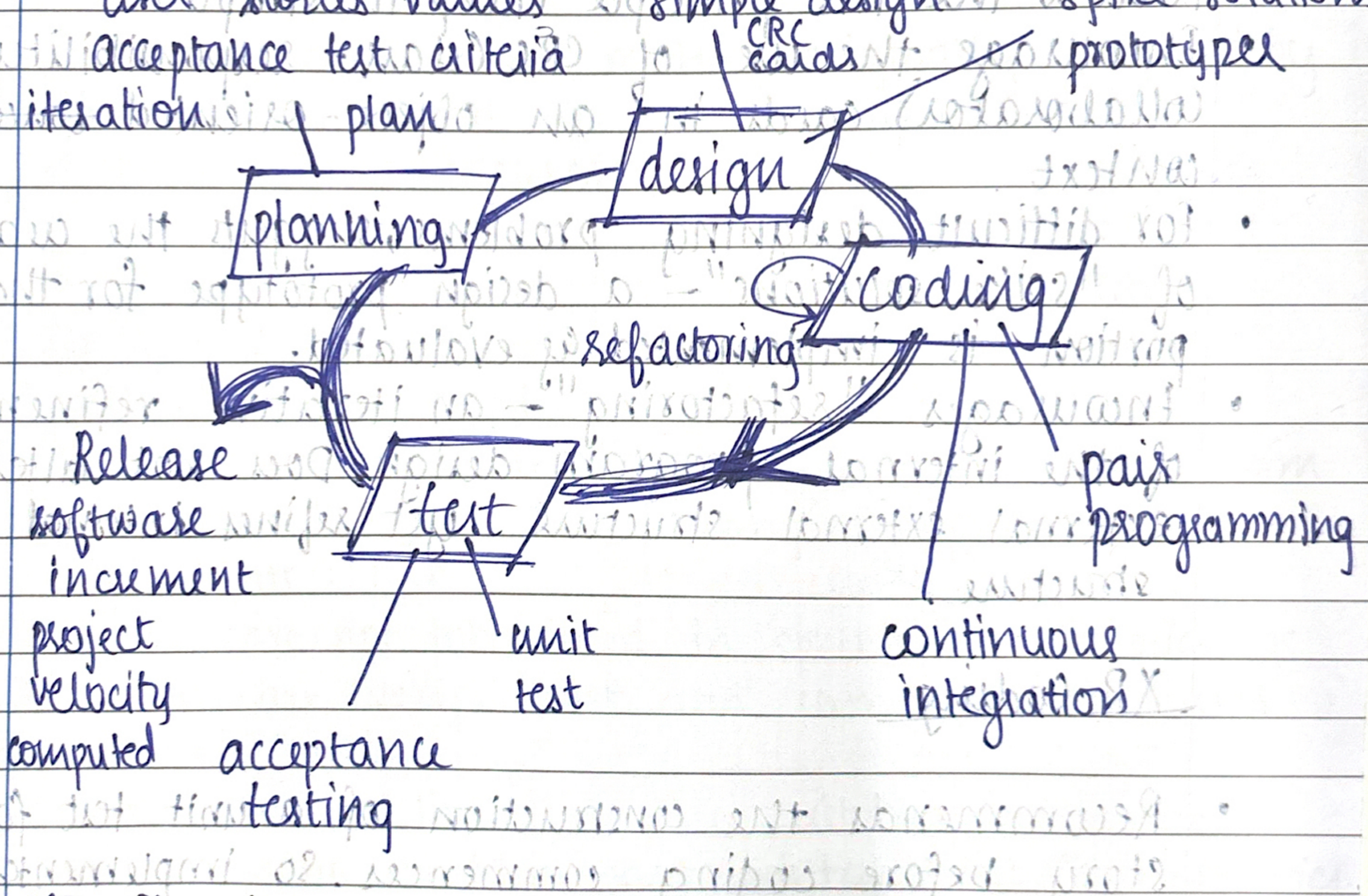
- Delivers multiple 'software increments', delivers an operational prototype / portion of an OS to collect customer feedback for adaptation

→ Agility ~~Process - I~~. Principles

(1) Our highest principles is to satisfy the customer through early and continuous delivery of valuable software
(2) Welcome changing requirements, even late in dev. Agile process harness change for the customer's competitive advantage.
(3) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
(4) Business people and developers must work together daily throughout the project
(5) Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
(6) The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
(7) Working software is a primary measure of progress
(8) Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

⑨ continuous attention to technical excellence and good design enhances agility.
⑩ Simplicity - The art of maximizing the amount of work not done - is essential.
⑪ The best architectures, requirements, and designs emerge from self-organizing teams.
⑫ At regular intervals, the team reflects on how to become more effective, then tunes & adjusts its behaviour accordingly.
(★① graph diagram at the end)

⑧ Extreme Programming (XP)



XP Planning:

- Begins with the listening, leads to the creation of "user stories" that describes required output, features and functionality. Customer assigns a value to each story.

- Agile team assesses each story and assigns a cost
- Working together, stories are grouped for a deliverable increment next release.
- A commitment is made.
- After the first increment, "project velocity" namely no. of stories implemented during the first release is used to help define subsequent delivery dates for other increments.

XP Design

- follow the KIS principle (keep it simple)
- Encourage the use of CRC (class - responsibility - collaborator) cards in an object-oriented context.
- for difficult designing problems, suggests the creation of "spike solutions" - a design prototype for that portion is implemented & evaluated.
- Encourages "refactoring" - an iterative refinement of the internal program design. Does not alter external external structure, yet refines internal structure.

XP Coding:

- Recommends the construction of a unit test for a story before coding commences. So implementes can focus on what must be implemented to pass the test.
- Encourages "pair programming". Two people work together at one workstation. Real time problem solving, real time review for quality assurance.

DOMS | Page No. 13
Date / /

XP Testing

- All unit tests are executed daily and ideally should be automated.
- Regression tests are conducted to test the current and previous components
- Acceptance tests are defined by the customer and executed to assess customer visible functionality.

Q7.

(c) SCRUM

→ Software development method originally proposed by Schwaber & Beedle

Distinguishing features:

- Development work is partitioned into "packets"
- Testing and documentation are on-going as the product is constructed
- Work units occur in "sprints" and is derived from a "backlog" of existing changing prioritized requirements
- Changes are not introduced in sprints but in backlogs
- Meetings are very short and sometimes conducted without chairs
- "Demos" are delivered to the customer with the time-box allocated. May not contain all functionalities. So customers can evaluate and give feedbacks.

⑮ Capability Maturity Model Integration (CMMI)

→ Defines each process area in terms of "specific goals" and the "specific practises" required to achieve these goals.

→ Specific goals ~~achieve these~~ establishes the characteristics that must exist if the activities implied by a process area are to be effective

→ Specific practises refine a goal into a set of process-related activities

CMMI level

- level 0 (Incomplete)
  Process are not performed or have not achieved all the goals eg objectives defined by the CMMI for level-1 capability

- level 1 (Performed)
  All specific goals are performed as per defined by CMMI.

- level 2 (Managed)
  - All level 1 citeria have been satisfied
  - In addition to level 1
    - → People doing the work have ~~additional~~ access to adequate resources to get job done
    - → Stakeholders are actively involved
    - → work tasks and products are monitored, controlled, reviewed and evaluated for conformance to process description

→ Level 3 (Defined)
- All level 2 criteria have been achieved
- In addition;
  - → management & engineering processes documented & standardized & integrated into organization-wide software process

→ Level 4 (Quantitatively Managed)
- All level 3 criteria have been satisfied
- Software processes and products are quantitatively understood
- Controlled using detailed measures & assessments

→ Level 5 (Optimized)
- continuous process improvement is enabled by quantitative feedback from the process and testing innovative ideas

Q10. Explain Software Engineering Process.

The SEP is a structured set of activities involved in the development, operation and maintanence of software systems. It acts as a framework that helps engineers manage complexity, ensure quality, and meet customer requirements effectively.

→ Key Elements of a Software Engineering Process:

1. Layered Technology structure
- Software Engineering is built on layered tech. foundation.

Date / /

① Quality focus (Bedrock layer):
Ensures continuous improvement and a commitment to delivering high-quality software.

② Process (Framework layer):
Defines the sequence of tasks and how they are managed to achieve goals.

③ Methods: (Technical "how-to" layer)
Specific practises for analysis, design, coding, testing and support.

④ Tools (Automation layer):
Software tools to support processes and methods efficiently.

2. Generic Process framework Activities

These are universal activities in all software processes:

① Communication - Interact with customers/stakeholders to gather requirements.

② Planning - Define resources, risks, timelines, & tasks

③ Modeling - Analyze and design the system.

④ Construction - Write code & perform testing

⑤ Deployment - Deliver the system & gather user feedback.

3. → Umbrella Activities.

These support the entire development lifecycle:

① Project Tracking & Control

② Format Technical Reviews

③ Quality Assurance

④ configuation Management

4. → Process Adaptability

- The process framework is flexible: it can be adopted based on:

→ project size & complexity
→ Stakeholder needs.