1. Define Software Engineering. Differentiate with CS & System Eng.

- **Software Engineering**: Application of a systematic, disciplined, quantifiable approach to development, operation, and maintenance of software
  SE Unit 1 part 1 Notes
  .
- **Computer Science**: Theoretical study of algorithms, data, and computation.
- **System Engineering**: Focuses on overall system (hardware + software + processes).
- **Difference**:
  - CS → theory,
  - SE → disciplined practice for software,
  - SysEng → integrates software + hardware.

---

2. Steps of Software Process Cycle:
a) Communication
b) Planning
c) Modelling
d) Construction
e) Deployment

SE unit 1 part 2 Notes

---

3. Define Agility & two disadvantages of Agile.

- **Agility**: Ability to adapt to changes quickly while maintaining software quality.
- **Disadvantages**:
  1. Needs customer availability & collaboration.
  2. Difficult to use for large, complex projects.

---

4. Four Advantages of Agile:

1. Customer satisfaction through rapid delivery.
2. Flexible to changes.
3. Encourages teamwork & communication.
4. Reduces risk with incremental releases.

---

5. Software Process Assessment & Types:

- **Definition**: Disciplined examination of an organization’s software process to identify strengths/weaknesses
  SE unit 1 part 2 Notes
  .
- **Types**:
  - **Self-assessment**: By internal team.
  - **Second-party**: Internal but supervised by external team.
  - **Third-party**: Independent external body.

---

6. Software Engineering as Layered Technology:

SE unit 1 part 2 Notes

- **Quality Focus** (bedrock).
- **Process** (framework layer).
- **Methods** (technical how-to).
- **Tools** (automated support like CASE).

---

7. When to use Incremental Model?

SE unit 1 part 2 Notes

- Requirements well understood.

- Customer demands early delivery.
- Project can be developed in modules.

---

8. Necessary Conditions for Evolutionary Models:

- Requirements not clear in early stage.
- Frequent change expected.
- Large projects with high risk.

---

9. Two Advantages & Disadvantages of Evolutionary Model:

SE unit 1 part 2 Notes

- **Advantages**:
    1. Early feedback from customer.
    2. Risk reduction by iterative refinement.
- **Disadvantages**:
    1. Costly due to repeated iterations.
    2. Not suitable for smaller projects.

---

10. Aspect Software Development Process (AOP):

- Aspect-Oriented Programming helps modularize cross-cutting concerns (e.g., logging, security).
- Provides better separation of concerns.

---

11. Justify – "Agile yesterday, today, tomorrow":

- **Yesterday**: Emerged as response to rigid models (Waterfall).
- **Today**: Widely adopted in IT industry.
- **Tomorrow**: Continues to evolve with DevOps, CI/CD, AI-driven agility.

---

12. When to use Agile Model?

- Requirements change frequently.
- Customer involved closely.
- Project requires rapid delivery.

---

13. Software Process Maturity Model:

SE unit 1 part 2 Notes

- **CMMI Levels:**
    1. Initial (ad hoc).
    2. Managed.
    3. Defined.
    4. Quantitatively Managed.
    5. Optimizing.

Perfect 👍 Let me expand each **long answer** so you get 5-mark worthy content, with a bit more explanation and examples from your notes.

📘 Long Questions (Detailed Answers)

---

11 (a) Explain Software Process Framework.

A **software process framework** is a structured set of activities, actions, and tasks carried out to build software. It defines *how* software is developed and maintained.

Five Framework Activities:

1. **Communication** – Interaction with stakeholders to gather requirements and understand objectives.
   Example: Interviewing a client before developing a fitness tracking app.

2. **Planning** – Estimating resources, defining schedules, and assessing risks.

3. **Modeling** – Creating system architecture, data models, and design workflows (e.g., UML diagrams).

4. **Construction** – Coding and testing the system modules.

5. **Deployment** – Delivering the product (incrementally or fully) to customers for use and feedback.

Umbrella Activities: (support every stage)

- Project tracking & control
- Risk management
- Software Quality Assurance (SQA)
- Configuration management (SCM)
- Technical reviews

✅ **Significance**: Provides a roadmap that ensures software is built systematically, reduces risk, and maintains quality.

---

11 (b) Explain Waterfall Process Model.

The **Waterfall model** is the earliest SDLC model, also known as the **classic life cycle model**. It is a **linear, sequential** approach.

Phases:

1. Requirements analysis

2. Design

3. Implementation (coding)

4. Testing
5. Deployment & Maintenance

Advantages:

- Simple and easy to understand.
- Works well when requirements are fixed and clear.
- Each phase has clear deliverables.
- Good for small, low-risk projects.

Disadvantages:

- Not flexible (changes are difficult once a phase is complete).
- Risk is high if requirements are unclear.
- Doesn't work well for complex or object-oriented projects.
- Progress is hard to measure within phases.

✅ **Example**: Good for projects like payroll systems where requirements rarely change.

---

12 (a) Compare Waterfall & Incremental Models.

Feature	Waterfall Model	Incremental Model
Approach	Linear, sequential	Iterative, modular
Flexibility	Rigid, no changes once phase starts	Flexible, supports changes in increments
Delivery	Entire system at the end	Partial system delivered in increments
Customer Involvement	Only at beginning & end	Continuous involvement
Risk Handling	High risk if requirements change	Lower risk, problems detected early

Best For	Small projects with fixed requirements	Large/modular projects needing early releases

✅ **Summary**: Incremental is more adaptive and customer-focused than Waterfall.

---

12 (b) Compare Prototyping & Spiral Models.

Feature	Prototyping Model	Spiral Model
Goal	Build a quick prototype to clarify requirements	Manage risks in large/complex projects
Approach	Trial-and-error with prototypes	Cyclic iterations (planning, risk analysis, prototyping, development)
Customer Role	Continuous feedback on prototype	Stakeholder involvement at each iteration
Risk Management	Limited	Strong (risk-driven model)
Cost	Cheaper for small/medium projects	Expensive, suited for large projects

✅ **Summary**: Prototyping is best for unclear requirements; Spiral is best for high-risk, large systems.

---

13 (a) Explain Evolutionary Process Model.

The **evolutionary model** combines **incremental** and **iterative** development. Software is developed in steps, refined repeatedly until the final product is achieved.

Types:

1. **Prototyping Model** – Build a working model for requirement validation.
2. **Spiral Model** – Iterative + risk management.
3. **Concurrent Model** – Activities (design, coding, testing) can run in parallel states.

Advantages:

- Handles changing requirements.

- Customer feedback at every stage.
- Risks identified and reduced early.

Disadvantages:

- Expensive due to multiple iterations.
- Difficult to manage with poor communication.

✅ **Use Case**: Complex, evolving applications like online banking systems.

---

13 (b) Explain Incremental Process Model.

The **incremental model** delivers the system in smaller **modules (increments)**. Each increment adds functionality until the full product is complete.

Process:

1. Collect initial requirements.
2. Develop the first increment → deliver to customer.
3. Gather feedback → refine & add next increment.
4. Repeat until full system is ready.

Advantages:

- Early delivery of working product.
- Easier to test/debug smaller modules.
- Cost effective.

Disadvantages:

- Needs good planning and design.
- Interfaces between modules must be well-defined.
- Customer must clearly know requirements.

✅ **Example**: Building an e-commerce site where the first increment is “product browsing,” second is “shopping cart,” third is “payment gateway.”

---

14 (a) Explain Generic View of Process.

A **generic process** includes essential activities for any software project:

- **Communication** (requirements with stakeholders)
- **Planning** (tasks, resources, risk estimation)
- **Modeling** (design architecture, data flow)
- **Construction** (coding & testing)
- **Deployment** (delivery & feedback)

Umbrella activities (apply across all phases): risk management, configuration management, technical reviews, and quality assurance.

✅ **Key Idea**: Provides a universal structure that can be adapted by different models like Agile, Waterfall, Incremental.

---

14 (b) Explain Unified Process Model (UP).

The **Unified Process (UP)** is an object-oriented, iterative, and incremental framework.

Phases:

1. **Inception** – Define project scope, business goals, high-level requirements.
2. **Elaboration** – Refine use cases, build architectural baseline, plan resources.
3. **Construction** – Actual coding, unit testing, component integration.
4. **Transition** – Deliver system to users, perform beta testing, training, documentation.
5. **Production** – Maintenance, updates, bug fixes.

Features:

- Use-case driven.

- Architecture-centric.
- Iterative & incremental.
- Supports large, object-oriented projects.

✅ **Example**: Used in Rational Unified Process (RUP) for enterprise-scale software.

---

15 (a) Explain Agile Process Model.

The **Agile Model** is an iterative, incremental approach emphasizing adaptability, collaboration, and customer satisfaction.

Principles:

- Deliver working software frequently.
- Welcome changing requirements.
- Promote collaboration between developers and customers.
- Prioritize simplicity and quick delivery.

Advantages:

- Highly flexible.
- Continuous feedback.
- Faster time-to-market.

Disadvantages:

- Needs customer availability.
- Difficult for large teams without discipline.

✅ **Use Case**: Web apps, mobile apps, or projects where requirements evolve rapidly.

---

15 (b) Explain RAD (Rapid Application Development).

The **RAD model** emphasizes **fast development** using component reusability, prototyping, and parallel development.

Phases:

1. **Communication** – Requirement workshops with users.
2. **Planning** – Multiple teams working in parallel.
3. **Modeling** – Business, data, and process modeling.
4. **Construction** – Use of pre-built components, prototyping, and iterative testing.
5. **Deployment** – Quick delivery for feedback.

Advantages:

- Faster development (2-3 months cycles).
- High user involvement ensures satisfaction.
- Increases reuse of components.

Disadvantages:

- Needs highly skilled developers.
- Requires expensive CASE tools.
- Not suitable for all projects (e.g., complex, large systems).

✅ **Example**: RAD works best for modular business applications like HR management or sales tracking systems.

👉 Do you want me to **make these long answers into a structured table + bullet points format (like exam notes)** or keep them as detailed paragraphs for descriptive answers?