📘 SOFTWARE ENGINEERING QUICK NOTES

✅ 1. SOFTWARE BASICS

Software Crisis

- Early projects: Late delivery, over budget, poor quality.
- Caused by poor planning, unclear requirements, lack of tools.

What is Software?

- Software = Programs + Documentation + Data.

Nature of Software

- Intangible
- Doesn’t wear out
- Easy to change, hard to manage

Defining Software

- Collection of related programs, procedures, and documents.

Software Application Domains

- **System Software** (e.g., OS)
- **Application Software** (e.g., MS Word)
- **Embedded Software** (e.g., firmware)
- **Web Apps**
- **AI Software**

Legacy Software

- Old, still in use
- Difficult to maintain but valuable

Changing Nature of Software

- **Web Apps**: Browser-based
- **Mobile Apps**: Portable, platform-specific

- **Cloud Computing**: Internet-based storage & services
- **Product Line Software**: Shared core + variations for different needs

Software Engineering

- Systematic, disciplined approach to software development
- Ensures quality, cost-efficiency, and maintainability

The Software Process

- Set of activities to develop software

Process Framework

- Common activities:
    - Communication
    - Planning
    - Modeling
    - Construction
    - Deployment

Umbrella Activities

- Project tracking & control
- Risk management
- Quality assurance
- Reviews & audits
- Configuration management
- Reusability management

Software Engineering Principles

- Understand the problem fully
- Plan before you build
- Reuse existing code
- Maintain quality control

✅ 2. SDLC AND PROCESS ACTIVITIES

Software Development Life Cycle (SDLC)

- Steps: Requirements > Design > Implementation > Testing > Maintenance

A Generic Process Model

- **Phases**:
    - Communication
    - Planning
    - Modeling
    - Construction
    - Deployment

Framework Activities

- Standard process structure applied to all software projects

Software Process Flow

- **Linear**, **Iterative**, **Evolutionary**, **Concurrent** flows

Task Sets

- Activities, milestones, work products for a framework activity

Process Patterns

- Reusable best practices for solving common problems

Process Assessment and Improvement

- Measure current process → Identify weaknesses → Improve

✅ 3. PROCESS MODELS

What is a Process Model?

- Structured way to develop software

Process Flow Types

- **Linear**: One phase after another
- **Iterative**: Repeat phases
- **Evolutionary**: Build, get feedback, improve
- **Concurrent**: Parallel development

Prescriptive vs Descriptive Models

- **Prescriptive**: Predefined process (e.g., waterfall)
- **Descriptive**: Real-world, adaptive processes

Prescriptive Process Models

- Follow a fixed structure

The Waterfall Model

- Sequential phases: Requirements → Design → Code → Test → Deploy
- Easy to manage, but rigid

Incremental Process Models

- Develop in increments, each adds more functionality
- Easier testing & feedback

Evolutionary Process Models

- Build a working version, improve with feedback
- Examples: Prototyping, Spiral Model

Concurrent Models

- Activities happen in parallel
- Useful in dynamic environments

✅ 4. SPECIALIZED PROCESS MODELS

Component-Based Development

- Build systems using reusable components

Formal Methods Model

- Based on mathematical specification
- Used in critical systems (e.g., banking, aviation)

Aspect-Oriented Software Development

- Separates core logic from cross-cutting concerns (e.g., logging, security)

Unified Process (UP)

- **Phases of UP**:
    - **Inception**: Define scope
    - **Elaboration**: Detailed planning
    - **Construction**: Build software
    - **Transition**: Deploy to users
- Object-oriented, iterative model

Brief History

- Unified Process created as a standardized OO development framework

---

✅ 5. PERSONAL & TEAM MODELS

Personal Software Process (PSP)

- Helps developers track and improve their own work quality

Team Software Process (TSP)

- PSP at team level: Encourages planning, tracking, and quality as a team

Here’s your **concise, exam-friendly notes** covering all your topics with key highlights. I’ve structured them for quick revision, focusing on definitions, types, and core concepts.

---

1. Software Crisis

- **Definition**: Early software projects failed due to poor planning, cost overruns, and unreliable products.

- **Causes**:

  - Unrealistic deadlines.

  - Poor requirements.

  - Lack of engineering practices.

2. Software & Its Nature

- **Software**: Instructions + data that make hardware functional.

- **Nature**:

  - **Intangible** (no physical form).

  - **Easily replicated**.

  - **Degrades** (not wears out).

- **Application Domains**:

  - **System software** (OS, drivers).

  - **Application software** (mobile apps, web apps).

  - **Embedded software** (IoT, smart devices).

- **Legacy Software**: Outdated but critical systems (e.g., old banking software).

3. Changing Nature of Software

- **Web apps** (dynamic, browser-based).

- **Mobile apps** (iOS/Android).

- **Cloud computing** (AWS, SaaS).

- **Product-line software** (reusable components for similar products).

4. Software Engineering

- **Definition**: Systematic approach to software development/maintenance.

- **Principles**:

  - Modularity, abstraction, anticipate change.

- **Process Framework**:

  - **Activities**: Communication, planning, modeling, construction, deployment.

  - **Umbrella Activities**: QA, risk management, documentation.

---

5. Software Development Life Cycle (SDLC)

- **Generic Model**:

  1. **Requirements** → 2. **Design** → 3. **Development** → 4. **Testing** → 5. **Deployment** → 6. **Maintenance**.

- **Process Patterns**: Reusable solutions to common problems (e.g., "Divide and Conquer").

---

6. Process Models

Prescriptive vs. Descriptive Models

- **Prescriptive**: Strict steps (Waterfall).

- **Descriptive**: Flexible (Agile).

Key Models

1. **Waterfall**: Linear, rigid phases.

2. **Incremental**: Deliver in chunks.

3. **Evolutionary** (e.g., **Spiral Model**): Iterative + risk analysis.

4. **Concurrent**: Parallel workflows (e.g., coding + testing overlap).

Specialized Models

- **Component-Based**: Reuse existing components.

- **Formal Methods**: Mathematical rigor (for safety-critical systems).

- **Aspect-Oriented**: Separates cross-cutting concerns (e.g., logging).

- **Unified Process (UP)**:

  - **Phases**: Inception, Elaboration, Construction, Transition.

Personal/Team Models

- **PSP (Personal)**: Focus on individual metrics.

- **TSP (Team)**: Scales PSP for teams.

---

7. Agile Development

Core Concepts

- **Agility**: Flexibility + customer collaboration.

- **Manifesto**:

  - Individuals > processes.

  - Working software > documentation.

- Customer collaboration > contract negotiation.

- Responding to change > following a plan.

User Stories

- **Template**: "As a [user], I want [feature] so that [benefit]."

- **Good Stories**: **INVEST** (Independent, Negotiable, Valuable, Estimable, Small, Testable).

- **Spike**: Time-boxed research task.

Agile Planning

- **Effort vs. Duration**: Effort = work hours; Duration = calendar time.

- **Estimation Styles**:

  - **Planning Poker**: Team votes on effort.

  - **Card Sorting**: Prioritize features.

- **Velocity**: Work completed per sprint (measures team capacity).

Agile Methods

1. **Scrum**:

   - **Roles**: Product Owner, Scrum Master, Team.

   - **Ceremonies**: Sprint Planning, Daily Standup, Retrospective.

2. **XP (Extreme Programming)**:

   - **Practices**: Pair programming, TDD, continuous integration.

3. **Kanban**: Visual workflow (columns: To Do → In Progress → Done).

4. **DSDM (Dynamic Systems)**: Focus on business needs.

5. **FDD (Feature-Driven)**: Feature-centric iterations.

Key Highlights for Exam

- **Waterfall vs. Agile**: Rigid vs. flexible.

- **User Stories**: INVEST criteria.

- **Scrum Ceremonies**: Sprint, Standup, Retro.

- **Velocity**: Measures team output.

Final Tips

1. **Memorize Agile Manifesto** (4 values).

2. **Know 1-2 process models in depth** (Waterfall, Scrum).

3. **Focus on definitions** (e.g., legacy software, spikes).

You've got this! Revise these bullet points, and you'll cover 80% of the paper. Good luck!
🚀

Alright! Let's make magic happen 🔥
Here's your **super-simplified, short, exam-saving notes** on all the topics from the image.
Perfect to revise quickly even at a wedding 💒📚

✅ 1. Software Basics

- **Software Crisis**: Problems faced in early software development — projects were late, over budget, and had bugs.
- **What is Software?**: Set of programs + documentation + data.
- **Nature of Software**: Intangible, doesn't wear out, easy to change but hard to manage.
- **Defining Software**: Software = Programs + Procedures + Documentation.
- **Software Application Domains**:
    - System Software
    - Application Software
    - Embedded Software
    - Web Apps
    - AI Software
- **Legacy Software**: Old, still in use, hard to maintain.
- **Changing Nature of Software**:
    - Web apps = online apps
    - Mobile apps = phone/tablet apps
    - Cloud = hosted over internet
    - Product-line software = shared core, different versions

---

✅ 2. Software Engineering Basics

- **Software Engineering**: Discipline to build quality software in budget + time.
- **The Software Process**: A structure followed to build software.

- **Process Framework** (3 main components):
    - **Tasks**
    - **Milestones**
    - **Work Products**
- **Umbrella Activities** (support activities):
    - Software project tracking
    - Risk management
    - Quality assurance
    - Documentation
- **Software Engineering Principles**:
    - Understand the problem
    - Plan before building
    - Keep it simple
    - Ensure quality

✅ **3. Software Development Life Cycle (SDLC)**

- **SDLC**: Step-by-step process to build software from idea to delivery.
- **A Generic Process Model**:
    - Communication
    - Planning
    - Modeling
    - Construction
    - Deployment
- **Framework Activities** = Core stages of SDLC.
- **Software Process Flow**: Flow of steps (Linear, Iterative, Evolutionary).
- **Task Sets**: Set of tasks, work products & milestones.

- **Process Patterns**: Common solutions to recurring problems.
- **Process Assessment & Improvement**: Analyze → Identify gaps → Improve.

✅ **4. Process Models**

- **What is a Process Model?**: A strategy to organize SDLC.
- **Process Flow Types**:
  - **Linear**
  - **Iterative**
  - **Evolutionary**
  - **Parallel**
- **Prescriptive vs Descriptive**:
  - **Prescriptive**: Fixed structure (e.g., waterfall)
  - **Descriptive**: Real-world, flexible process

✅ **5. Prescriptive Process Models**

- **The Waterfall Model**:
  - Linear stages: Requirements → Design → Coding → Testing → Maintenance
- **Incremental Models**:
  - Software built in small parts/releases
- **Evolutionary Models**:
  - Built, feedback taken, improved (e.g., Prototyping, Spiral)
- **Concurrent Models**:
  - Activities done in parallel (e.g., design & coding together)

✅ **6. Specialized Process Models**

- **Component-Based Development**: Reuse existing software components.

- **Formal Methods Model**: Uses mathematical proofs (for critical systems).
- **Aspect-Oriented Development**: Separates core logic and secondary concerns (like security/logging).
- **Unified Process (UP)**:
    - **Phases**:
        - Inception
        - Elaboration
        - Construction
        - Transition
    - Iterative + Object-Oriented

✅ 7. Personal and Team Models

- **Personal Software Process (PSP)**: Helps developers improve their work individually.
- **Team Software Process (TSP)**: Team-level version of PSP to improve team performance.

📌 Quick Tip to Remember Model Types:

WIECEU = Waterfall, Incremental, Evolutionary, Concurrent, Unified

---

Let me know if you want a 1-pager PDF of these notes or a quiz to revise quickly!
You've totally got this — just focus on these key points, and you'll be able to handle the paper like a boss 😎📖