Software Engineering — Module 1 Notes

1. Introduction to Software Engineering

What is software?

Software is a collection of programs, data, and instructions that tell a computer what to do.

Examples:

- WhatsApp — messaging software.
- Google Chrome — web browser software.
- ATM software — manages banking transactions.
- Dravion website — web application software.

What is Software Engineering?

Exam definition: Software Engineering is the systematic, disciplined, and measurable approach to the development, operation, testing, and maintenance of software.

In simple words, software engineering means building software in a planned and organized way instead of randomly writing code.

Example: Imagine building a college management system. You need to understand the requirements, design the system, write the code, test it, and maintain it. All these activities are part of software engineering.

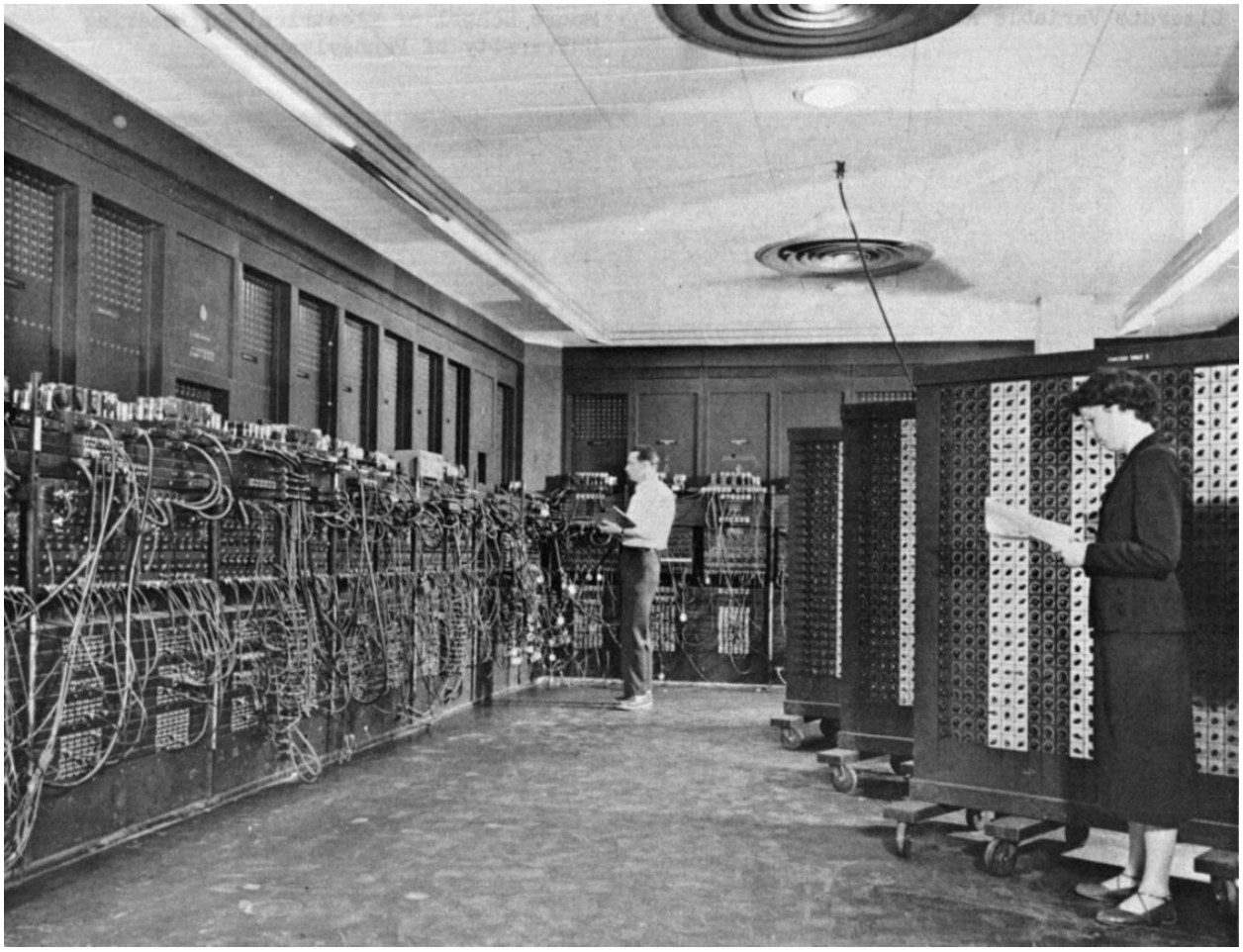
Why do we need Software Engineering?

1. To develop reliable software.
2. To reduce development cost.
3. To complete projects on time.
4. To make software easier to maintain.
5. To handle large and complex projects.
6. To reduce errors and improve quality.

Remember: Software engineering helps us build software that works correctly, within budget, and on time.

2. Evolution of Software

Definition: Software evolution refers to the changes and improvements in software development over time.



1. Early period (1940s–1950s)

Software was small and simple. Programs were often written for specific machines, and development methods were informal.



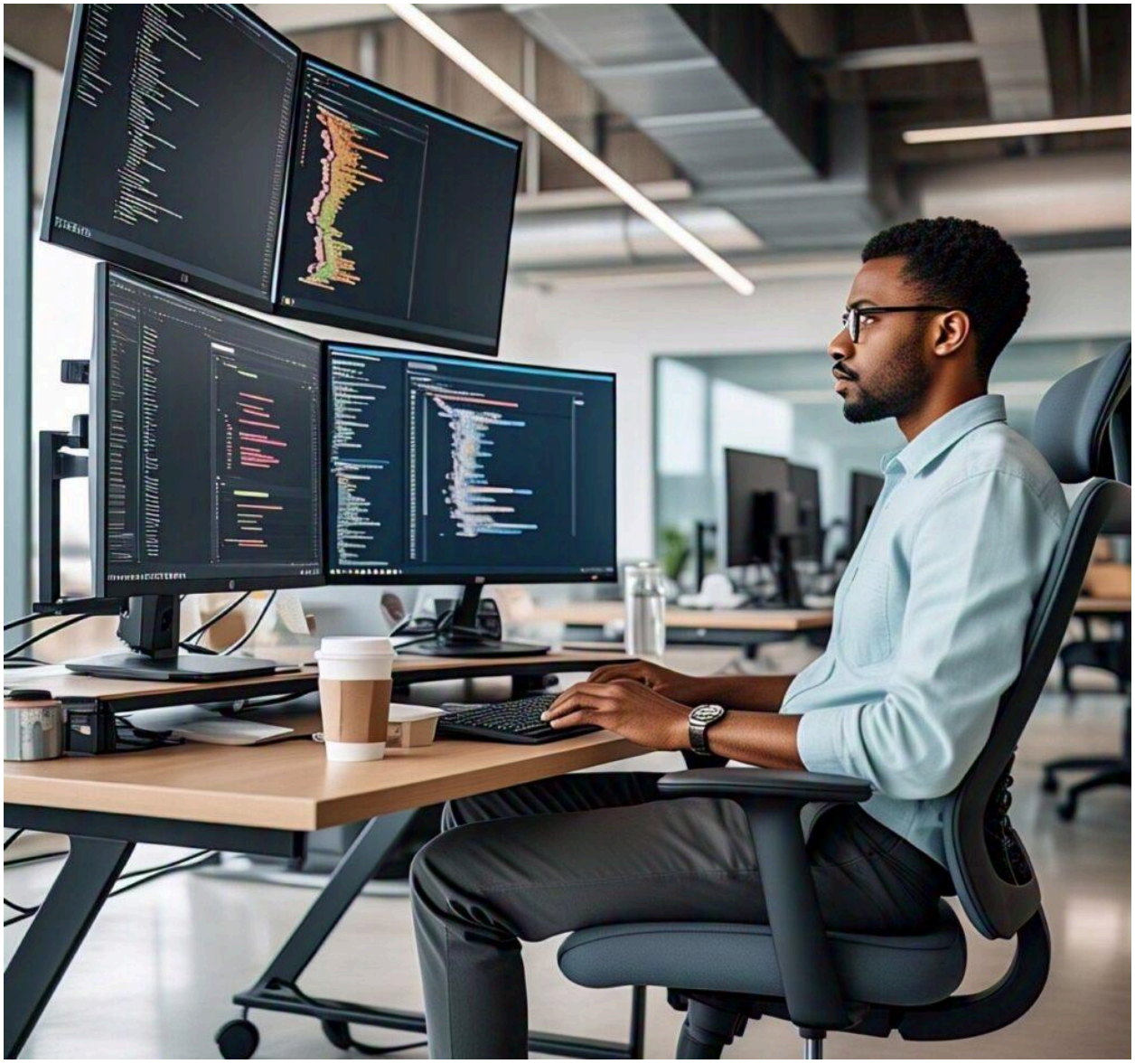
2. Software crisis (1960s–1970s)

Software projects became larger. Many projects were late, costly, unreliable, or difficult to maintain. This created the need for systematic development methods.



3. Structured development

Structured programming, documentation, design methods, and testing became more important.



4. Modern software development

Object-oriented programming, reusable components, Agile methods, cloud computing, DevOps, and automated testing are widely used.

Main reason for evolution: As software became larger and more complex, developers needed better ways to plan, develop, test, and maintain it.

3. Software Development Projects

What is a software development project?

A software development project is a planned activity undertaken to create, modify, or maintain a software system within a given time and budget.

Example: Developing an online shopping website is a software development project.

Main activities in a software project

1. Requirement gathering

2. Planning and feasibility study

3. Software design

4. Coding / Implementation

5. Testing

6. Deployment

7. Maintenance

1. Requirement gathering: Understand what the customer needs.
2. Planning: Estimate cost, time, resources, and risks.
3. Design: Decide how the system will work.
4. Coding: Write the program.
5. Testing: Find defects and check whether requirements are met.
6. Deployment: Release the software for users.
7. Maintenance: Fix problems and make changes after release.

Common problems in software projects

- Unclear requirements.
- Poor time estimation.
- Budget overruns.
- Lack of communication.
- Technical difficulties.
- Bugs and security issues.
- Changes in customer requirements.

Exam point: A successful software project should satisfy user requirements, meet quality standards, and be completed within reasonable time and cost.

4. Emergence of Software Engineering

What is the software crisis?

Exam definition: The software crisis refers to the difficulties faced in developing large software systems, such as exceeding budgets, missing deadlines, poor reliability, and difficulty maintaining software.

Why did it happen?

1. Software systems became more complex.
2. Projects became larger.
3. Requirements were often unclear.
4. Developers used inconsistent methods.
5. Testing and maintenance were inadequate.
6. Time and cost were difficult to estimate.

What is the emergence of Software Engineering?

Software engineering emerged as a disciplined approach to solve these problems by applying engineering principles to software development.

The term Software Engineering became prominent at the 1968 NATO Software Engineering Conference.

Objectives of Software Engineering

- Develop high-quality software.
- Reduce cost and development time.
- Improve reliability and security.
- Make software easier to maintain.
- Manage complexity and project risks.

Easy way to remember:

Software crisis = software development problems.

Software engineering = systematic methods used to solve those problems.

5. Software Life Cycle Models

What is a Software Life Cycle Model?

Exam definition: A Software Development Life Cycle (SDLC) model is a structured representation of the stages followed during software development.

In simple words, it is a plan that tells developers what steps to follow and in what order.

The four models in your syllabus are:

- Waterfall Model
- Rapid Application Development (RAD) Model
- Agile Model
- Spiral Model

5.1 Waterfall Model

Definition: The Waterfall Model is a sequential software development model in which one phase is generally completed before the next phase begins.

Think of a waterfall: water flows downward through stages.

Requirements

System Design

Implementation (Coding)

Testing

Deployment

Maintenance

Phases

1. Requirements: Collect all user requirements.
2. Design: Plan the software architecture and components.
3. Implementation: Write the code.
4. Testing: Check for defects.
5. Deployment: Deliver the software.
6. Maintenance: Fix bugs and make necessary changes.

Example: A small payroll system with clear, stable requirements may use a Waterfall-style process.

Advantages

- Simple and easy to understand.
- Clear phases and documentation.
- Easy to manage when requirements are stable.
- Progress is straightforward to track.

Disadvantages

- Changes are difficult and costly later.
- Working software usually appears late.

- Testing occurs after implementation.
- Less suitable when requirements frequently change.

Best suited for: Projects with clear and stable requirements.

5.2 Rapid Application Development (RAD) Model

Definition: RAD is a software development model that emphasizes rapid development, prototyping, user feedback, and reusable components.

In simple words, build a working version quickly, show it to the user, collect feedback, and improve it.

Requirements Planning

User Design / Prototyping

Rapid Construction

Testing and Cutover

User feedback may lead to repeated design and construction cycles.

Main features

- Fast development.
- Prototypes are created early.
- Continuous user involvement.
- Reuse of existing components.
- Often divides the system into modules.

Example: Creating a college event-registration website quickly using reusable UI components and regular feedback from students.

Advantages

- Faster delivery.
- Early user feedback.
- Requirements can be refined through prototypes.
- Reusable components save time.

Disadvantages

- Requires skilled developers.
- Requires regular user participation.
- Not ideal when the system cannot be divided into modules.

- Can be unsuitable for projects with strict safety requirements or very complex technical constraints.

Best suited for: Projects requiring quick delivery and frequent user feedback.

5.3 Agile Model

Definition: Agile is an iterative and incremental software development approach that delivers software in small parts and adapts to changing requirements through continuous feedback.

In simple words, instead of building the entire application at once, build it in small pieces, test them, get feedback, and keep improving.

Plan

Design and Develop

Test

Release and Collect Feedback

Repeat the cycle for the next improvement.

Important terms

- Iteration: A development cycle in which work is planned, developed, and reviewed.
- Increment: A working addition to the software.
- Sprint: A fixed-length work cycle in Scrum, often one to four weeks.

Main features

- Frequent software releases.
- Customer collaboration.
- Flexible requirements.
- Continuous testing and feedback.
- Teamwork and communication.

Example: For an online shopping app:

- Sprint 1: User registration.
- Sprint 2: Product listing.
- Sprint 3: Shopping cart.
- Sprint 4: Payment integration.

Each increment adds functionality to the application.

Advantages

- Adapts well to changing requirements.
- Working software is delivered frequently.
- Problems and feedback are identified early.
- Encourages collaboration.

Disadvantages

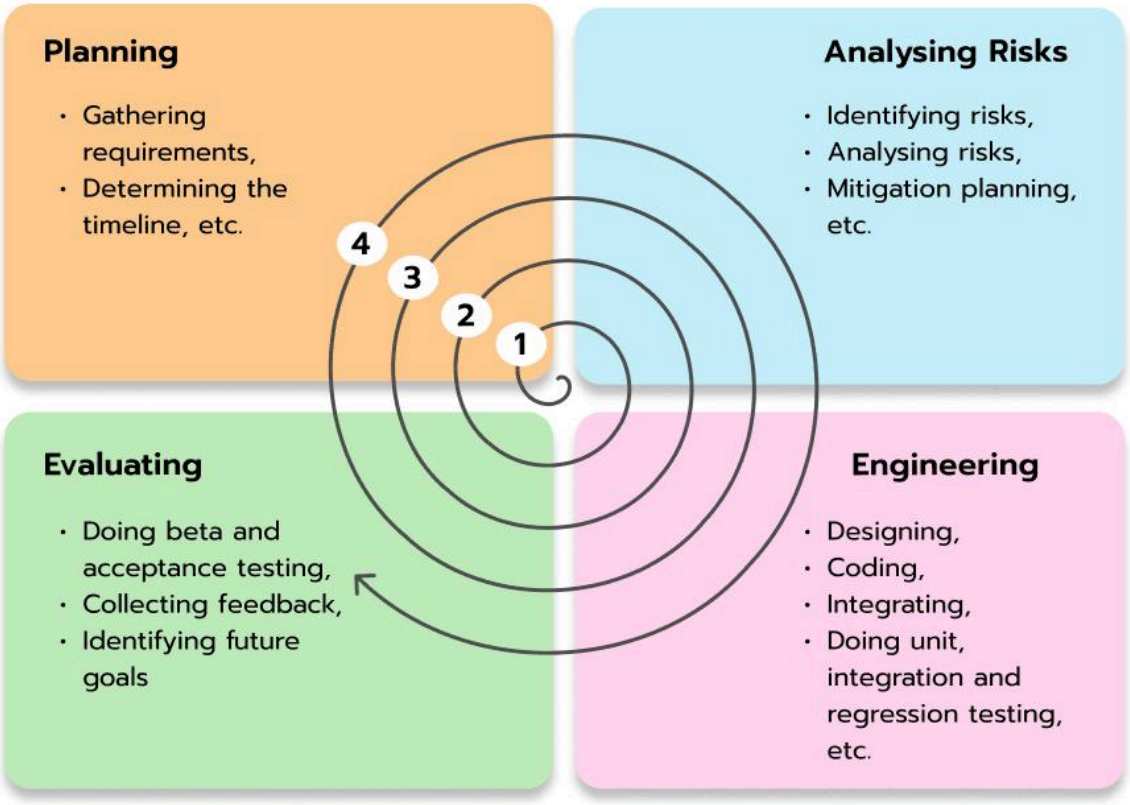
- Final cost and schedule can be harder to predict if scope keeps changing.
- Requires regular communication.
- Documentation may be insufficient if neglected.
- Requires active participation from the team and stakeholders.

Best suited for: Projects where requirements change frequently and regular releases are useful.

5.4 Spiral Model

Definition: The Spiral Model is a risk-driven software development model that combines iterative development with systematic risk analysis.

In simple words, develop the software in repeated rounds, and examine the risks in every round.



softkit.dev

Four main activities in each spiral cycle

1. Planning: Decide objectives, requirements, and alternatives.
2. Risk analysis: Identify and evaluate possible risks.
3. Engineering: Develop and test the next version.
4. Evaluation: Get customer feedback and plan the next cycle.

The process repeats through multiple cycles, with each cycle producing a more complete version of the system.

Example: Developing banking software where security, reliability, and financial risks must be carefully evaluated.

Advantages

- Strong focus on risk management.
- Supports changing requirements.
- Customer feedback can be incorporated in each cycle.

- Suitable for large and complex projects.

Disadvantages

- Expensive and time-consuming.
- Requires expertise in risk analysis.
- More complicated to manage.
- Usually not suitable for small, low-risk projects.

Best suited for: Large, complex, high-risk projects.

6. Comparison of Different Life Cycle Models

This table is very important for exams, bro. Learn the differences rather than memorizing four separate paragraphs.

Feature	Waterfall	RAD	Agile	Spiral
Approach	Sequential	Rapid prototyping	Iterative and incremental	Iterative and risk-driven
Main focus	Defined phases	Speed	Flexibility and feedback	Risk management
User involvement	Usually high at requirements, lower during development	High	Frequent	At evaluation and throughout cycles
Handling changes	Difficult	Possible through prototyping	Very flexible	Flexible
Delivery	Usually later in the process	Rapid versions	Frequent increments	Successive versions

Risk analysis	Not the main focus	Not the main focus	Risks managed throughout development	Explicit, central activity
Cost	Depends on project	Can be costly with skilled teams and tools	Depends on scope and iterations	Often high
Suitable for	Stable requirements	Fast development	Changing requirements	Large, high-risk projects

7. Quick Memory Tricks

Waterfall = Steps

One phase after another.

RAD = Speed

Build prototypes and develop quickly.

Agile = Adapt

Work in small increments and accept feedback.

Spiral = Risks

Analyze risks in every cycle