## What is software? What is software engineering?

**Software** is a set of programs, data, and instructions that tells a computer how to perform specific tasks. It includes application software, which helps users perform activities, and system software, which manages hardware and provides services for applications.

**Software engineering** is the systematic, disciplined, and measurable approach to the development, operation, and maintenance of software. It focuses on producing reliable, efficient, scalable, and maintainable software products within time and budget.

## What is SDLC? Write down the principles that need to be followed during Software Development life cycle.

The Software Development Life Cycle (SDLC) is a systematic process used in software engineering to plan, design, develop, test, deploy, and maintain software.

**Purpose:**

- To produce high-quality software
- Delivered on time and within budget
- That meets or exceeds customer requirements

**Requirements Engineering:** Clearly understand and document what to build through user needs, feasibility, and specifications.

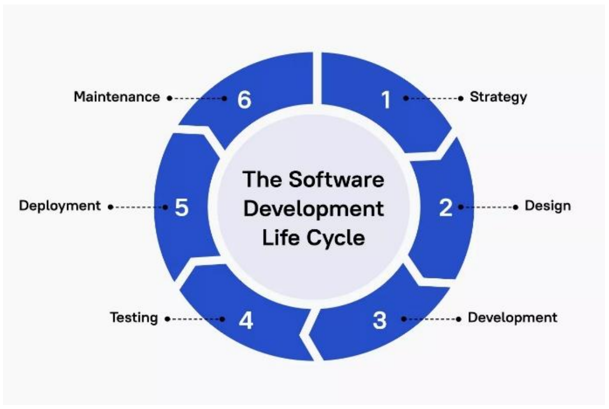
**Design:** Develop system architecture including modules, data flow, and user interfaces before coding.

**Development (Coding):** Implement the design using appropriate programming languages and tools.

**Software Quality Assurance (SQA):** Perform testing, verification, and validation to ensure correctness, reliability, and quality.

**Deployment:** Deliver the software to real users in the production environment for practical use.

**Maintenance:** Provide updates, bug fixes, security patches, and continuous support to improve and extend the software's life



## What is umbrella activity? Write down those activities.

Umbrella activities are **supporting tasks** that run across all phases of the Software Development Life Cycle (SDLC).

**Typical Umbrella Activities (in simple words):**

**Project Tracking & Control:** Monitor project progress, compare with plan, and take action on issues.

**Risk Management:** Identify risks and plan strategies to minimize them.

**Quality Assurance:** Ensure software meets quality standards through proper testing.

**Technical Reviews:** Review work products (design, code, docs) to catch errors early.

**Measurement:** Collect data (time, errors, cost) to improve process and product.

**Configuration Management:** Control changes in code, documents, and versions.

**Reusability Management:** Promote reuse of existing modules and components.

**Work Product Preparation:** Prepare reports, models, logs, and documentation.

## Write the comparison between this software modes named: Waterfall, V, Incremental, Spiral models. At least 6 points with aspects.

| Aspects | Waterfall Model | V Model | Incremental Model | Spiral Model |
|---|---|---|---|---|
| Development Flow | Linear, step-by -step | Linear + Verification / Validation | Small increments | Cyclic, iterative + Risk |
| Flexibility | Very rigid | Rigid but early test | Medium Flexible | Very flexible |
| Risk | Poor handling | Limited | Medium | Strong risk control |
| Testing | End only | Each stage | Per increment | Every cycle |
| Deployment | One final | Tested in steps | Partial working versions | Version after each spiral |
| Best Use in Project | Small, fixed requirements | Safety - critical | Medium to large scale | Big, risky and unclear requirements |
| Software exp. | Student Management System | Hospital Management System | E-commerce Platform | Banking Management System |

## Software Engineering Models:

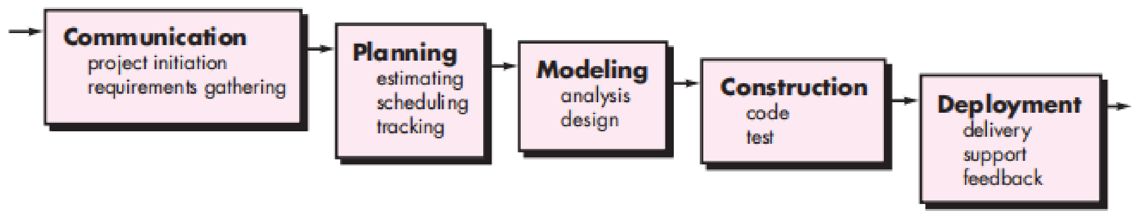
### Waterfall Model

The Waterfall Model is a step-by-step process that starts with requirement collection and moves through planning, design, coding, testing, and deployment. It is the oldest and most traditional software development model. The process is rigid and works best when requirements are fixed from the beginning.

**Example:** A small student project (e.g., Library Management System) where requirements are fixed and well understood.

**Diagram:**

FIGURE 2.3 The waterfall model



---

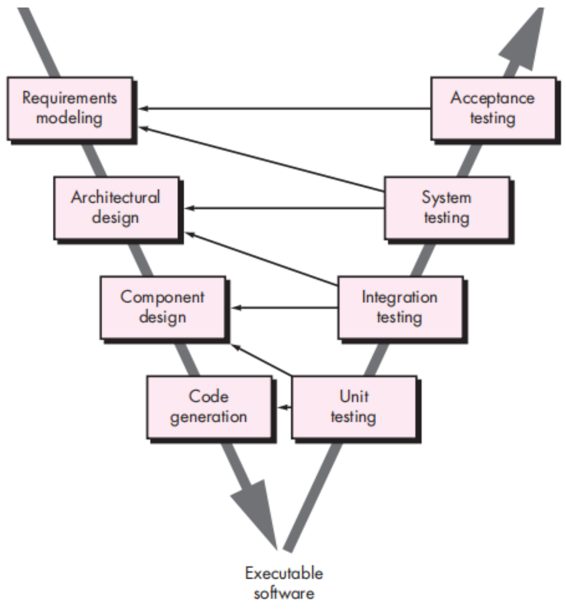
### ❖ V-Model

The V-Model is a variation of the Waterfall that adds testing at every stage. As the team goes down the "V," requirements and design are created, and while going up, testing is done to validate each stage. It mainly focuses on verification and validation, making it suitable for safety-critical projects.

**Example:** Medical or aerospace software where correctness and safety are critical (e.g., pacemaker control system).

**Diagram:**

The V-model



## ❖ Incremental Model

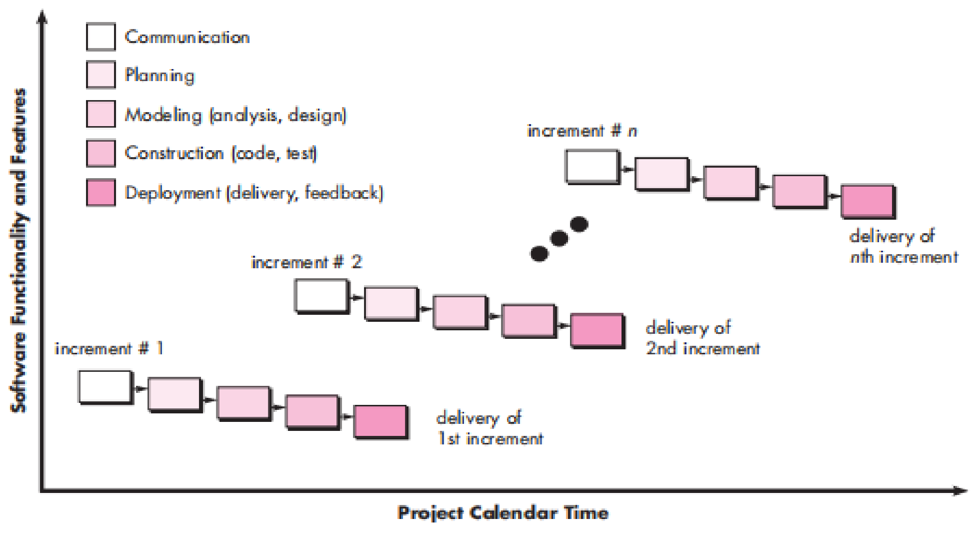
The Incremental Model develops software in small parts called increments. The first increment delivers a core product, and later increments add more features step by step. Customers can use early versions and give feedback for improvements, making it useful when requirements may change.

**Example:** E-commerce website where features (login, cart, payment, tracking) can be developed in increments.

**Diagram:**

FIGURE 2.5

The incremental model



Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
Software Functionality and Features
increment # 1
delivery of 1st increment
increment # 2
delivery of 2nd increment
increment # n
delivery of nth increment
Project Calendar Time

## ❖ Spiral Model

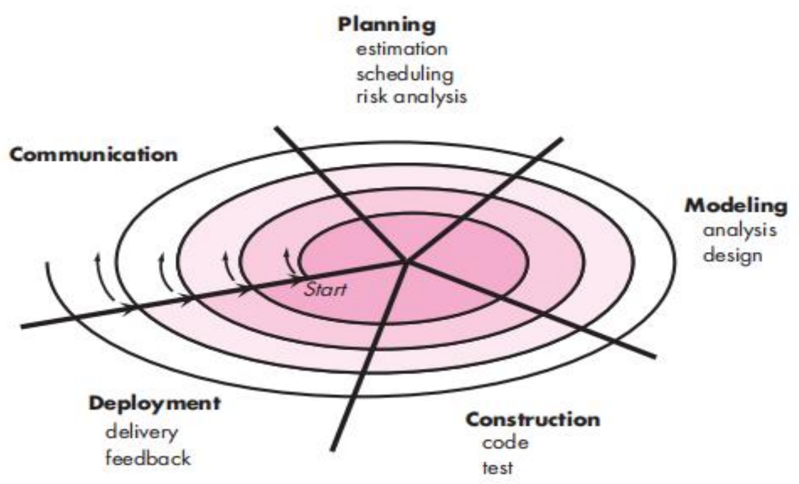
The Spiral Model is a risk-driven, cyclic development process. It grows the system gradually while reducing risks in each cycle. The model emphasizes risk analysis and stakeholder approval at milestones, making it ideal for large, high-risk, and complex projects.

**Example:** Large banking software or defense project where requirements are unclear, and risks must be managed carefully.

**Diagram:**

FIGURE 2.7

A typical spiral model



## Some Scenarios and use case analysis of software models and SDLC:

**Q1. A university wants to develop an online course registration system for students. As a software engineer, explain how you would handle this project using the SDLC.**

**Answer:**

**Feasibility Study:** Check if university has infrastructure (servers, internet, database).

**Requirement Analysis**: Collect requirements → student login, course add/drop, payment.

**Design:**

- Database design (tables for students, courses, enrollments).
- Interface design (student portal, admin panel).

**Implementation:** Develop web app using suitable technology (e.g., PHP/Java/Python).

**Testing:** Unit testing, integration testing, usability testing with students.

**Deployment:** Launch in phases (pilot with one department first).

**Maintenance:** Handle bugs, add new features (e.g., mobile support).

**Q2. A hospital wants to computerize patient records that are currently stored in paper files. Which software process model would you suggest and why?**

**Answer:**

**Suggested Model**: Incremental Model.

**Reason:** Hospital cannot wait for full system; urgent modules (patient registration, doctor appointment) should come first. Later increments can add billing, pharmacy, lab reports.

**Advantages:** Faster delivery of critical features, flexible to requirement changes.

Steps:

First increment: Patient registration + appointments.

Second increment: Billing + payment.

Third increment: Pharmacy + lab.

**Q3. An e-commerce startup wants to quickly launch a shopping website to beat competitors. Which SDLC model is suitable and how would you apply it?**

**Answer:**

**Suitable Model**: Prototyping Model

**Reason:** Startup needs speed, unclear requirements, need user feedback.

Steps:

- Develop prototype (basic product catalog + shopping cart)
- Get user feedback from customers.
- Refine prototype → add payment gateway, delivery tracking.
- Final system evolved after multiple iterations.

**Q4. A bank wants to build a mission-critical software for online transactions. Which process model will you choose and how will you ensure reliability?**

**Answer:**

**Suitable Model:** Waterfall Model (or V-Model).

**Reason:** Banking system must be highly reliable and secure; requirements are stable.

Steps:

1. **Requirement Analysis**: Document all functional + security needs.
2. **System Design:** Database for accounts, encryption for security.
3. **Implementation:** Strict coding standards, modular design
4. **Testing:** Rigorous testing (unit, integration, security testing).
5. **Deployment:** Controlled rollout with backup.
6. **Maintenance:** Regular updates, security patches.

**Q5. A government office wants to digitize all citizen records, but requirements are unclear and may change frequently. How would you handle it?**

**Answer**

**Suitable Model:** Spiral Model.

**Reason:** High risk project, changing requirements, long-term investment.

Steps:

- **Iteration 1:** Risk analysis, feasibility, prototype small database.
- **Iteration 2**: Expand features (citizen ID, addresses).
- **Iteration 3:** Add services (licenses, certificates).
- **Iteration 4:** Full-scale rollout.
  **Benefits:** Handles risks, flexible to change, progressive refinement.

**1. Waterfall Model**

The Waterfall model is a **linear and sequential software development model** where each phase (requirement, design, implementation, testing, deployment, and maintenance) must be completed before moving to the next. It is suitable for projects with **well-defined and stable requirements**.

**Examples:**

- ✓ Banking transaction system where rules and processes are fixed.
- ✓ Payroll system where salary calculation logic is predefined.
- ✓ Library management system with simple, clear requirements.

**Reason:**

It is best when the requirements are clear from the beginning and unlikely to change. It ensures discipline and structure in the development process.

---

**📌 2. V-Model (Verification and Validation Model)**

The V-Model is an extension of the Waterfall model that emphasizes **testing at every development stage**. Each development activity has a corresponding testing activity. This makes it suitable for **safety-critical and highly reliable systems**.

**Examples:**

- ✓ Hospital patient monitoring systems where accuracy is critical.
- ✓ Flight control software for airlines.
- ✓ Medical device software such as ECG or blood pressure monitors.

**Reason:**

It ensures early detection of errors and guarantees high quality and reliability, which is essential for life-critical applications.

---

**📌 3. Incremental Model**

The Incremental model develops a system in **small parts (increments)**. Each increment delivers a working product that can be used immediately, and new features are added in subsequent increments.

**Examples:**

- ✓ E-commerce websites where product listing is developed first, then cart, then payment.
- ✓ Hospital management systems starting with patient registration, then doctor appointments, then billing.
- ✓ University portals where admission features come first, followed by results and payments.

**Reason:**

It allows faster delivery of important features, early user feedback, and flexibility to adapt to changing requirements.

---

## 📌 4. Spiral Model

The Spiral model combines iterative development with **risk analysis at every cycle**. Each iteration includes planning, risk assessment, development, and evaluation. It is used for **large, complex, and high-risk projects**.

**Examples:**

- ✓ Military defense systems where risks are very high.
- ✓ Large banking systems where errors can cause huge financial losses.
- ✓ Research and development projects using new technology.

**Reason:**

It reduces risks through continuous evaluation, makes it easier to manage complexity, and allows gradual refinement of requirements.

---

## 📌 5. Prototype Model

The Prototype model builds a **working prototype first** to collect user feedback. The prototype is then refined repeatedly until the final system is developed. This is best for **unclear or evolving requirements**.

**Examples:**

- ✓ ATM user interface design for customer experience testing.
- ✓ Hospital appointment booking app where users can test and give feedback.
- ✓ Mobile applications where UI/UX plays a major role.

**Reason:**

It helps users visualize the system early, clarifies requirements, and reduces misunderstandings between clients and developers.

---

## 📌 6. **Evolutionary Model**

The Evolutionary model develops the system **step by step**, where each version is a working system that gradually evolves by adding new features and improvements. Users can give feedback after each version, and the system becomes more refined over time. This is best for projects with changing or partially known requirements.

**Examples:**

- ✓ Banking system: first release covers money transfer, later add bill payment, loan management.
- ✓ E-learning platform: start with course access, later add video lectures, discussion forums, online exams.
- ✓ Social media app: initial version with profile and posts, later add chat, video call, marketplace features.

**Reason:**

It allows gradual delivery of a usable system, accommodates changing requirements, collects user feedback continuously, and reduces risks of building a system that does not meet user needs.

📌 **Overall Bangla Explanation**

Waterfall model use hoy jekhane requirement ekdom fixed, V-model use hoy jekhane life-critical or high accuracy system dorkar, Incremental model use hoy jekhane step by step feature add korte hoy, Spiral model use hoy jekhane project boro, complex ebong risk beshi, ar Prototype model use hoy jekhane requirement clear na and client feedback dorkar.

Simple vabe bolte gele –

- ✓ **Waterfall → stable requirement project**
- ✓ **V-Model → accuracy and safety project**
- ✓ **Incremental → step by step feature build**
- ✓ **Spiral → risky and complex project**
- ✓ **Prototype → unclear requirement project**
- ✓ **Evolutionary → changing or partially known requirement project**

## Cholo ami 6 ta main model er jonno identifying terms diye dichi:

---

### 1. Waterfall Model

👉 Keywords jodi thake:

- "Clear and fixed requirements"
- "Step by step / linear / sequential process"
- "No changes allowed later"
- "Simple, small project"
- "Stable system (banking, payroll, calculator)"

**Best hint:** *Everything is known from start, changes not expected.*

---

### 2. V-Model

👉 Keywords jodi thake:

- "Verification and Validation"
- "Testing at each stage"
- "Safety-critical system"
- "Reliability must be 100%"
- "Medical devices / aviation / defense software"

**Best hint:** *Testing importance + safety critical project.*

---

**3. Incremental Model**

☞ Keywords jodi thake:

- “Quick delivery in parts/increments”
- “Basic version first, later add features”
- “Urgent modules needed first”
- “Hospital system, banking services, education platform”
- “Medium/large project, requirements change moderately”

**Best hint:** *Step by step delivery, usable product from first increment.*

---

**4. Spiral Model**

☞ Keywords jodi thake:

- “High risk project”
- “Risk analysis, risk handling”
- “Requirements unclear, long-term project”
- “Iterative + flexible”
- “Government project, big investment project”

**Best hint:** *Risk management + large complex project.*

---

**5. Prototype Model**

☞ Keywords jodi thake:

- “Unclear requirements”
- “User not sure what they want”
- “Demo or mock-up for feedback”
- “Interface prototype, dummy version first”
- “Startup project, new idea testing”

**Best hint:** *Prototype = show demo, refine, then final.*

---

**6. Evolutionary Model**

☞ Keywords jodi thake:

- “System grows step by step”
- “Product evolves gradually”
- “Changing requirements with working software at each stage”
- “Banking/social media/e-learning system”
- “Users give feedback → new features added continuously”

**Best hint:** *Continuous growth of real usable product.*

---

## 📌 Overall Bangla Shortcut

- **Waterfall:** shuru thekei clear → ek line e sesh.
- **V-Model:** testing beshi important + safety critical.
- **Incremental:** fast delivery part by part.
- **Spiral:** boro project + risk management.
- **Prototype:** requirements clear na → demo first.
- **Evolutionary:** usable product gradually evolve kore.

## 📌 White Box Testing

**Definition:**

White box testing (also called structural or glass-box testing) is a software testing method where the **internal structure, logic, and code** of the program is tested. The tester must know the programming and logic used.

**Example:**

✓ Checking all paths in a login function to ensure every if-else condition is correct.
✓ Testing loops to confirm they terminate correctly.
✓ Unit testing of individual functions in a program.

---

## 📌 Black Box Testing

**Definition:**

Black box testing is a method where the tester focuses only on the **input and output** of the software without knowing its internal code or logic. The tester treats the system as a "black box."

**Example:**

✓ Testing a login page by entering valid and invalid username/password combinations.
✓ Checking if an ATM dispenses the correct amount when requested.
✓ Verifying if the shopping cart updates correctly when adding/removing products.

---

## 📌 Comparison Between White Box and Black Box Testing

| Aspect | White Box Testing | Black Box Testing |
|---|---|---|
| **Knowledge Needed** | Requires knowledge of code and logic | No code knowledge needed, only requirements |
| **Focus** | Internal structure, code paths, conditions | External behavior, input-output |
| **Tester** | Usually done by developers | Usually done by testers or end-users |
| **Coverage** | Tests code coverage (paths, branches, loops) | Tests functionality as per requirements |
| **Example** | Testing loop conditions in a function | Testing login with valid/invalid inputs |
| **Best For** | Finding logical/code-level errors | Finding functional and user-level errors |

---

## 📌 Overall Bangla Explanation

White box testing holo jekhane developer code er vitore ghushe check kore – kon path kaj kortese, kono logical error ase naki. Ar black box testing holo jekhane user sudhu input dey ar output check kore – vitore ki vabe kaj korse seta jane na.

- ✓ White box = **"andarer code check"**
- ✓ Black box = **"baire theke input-output check"**