

# Software Development Methods

## Textbook

# Software Development Methods



Just like there are different ways to build a house, such as building everything at once or building in smaller, testable sections, there are different ways to create software. These approaches are called software development methodologies. They help teams organize their work, manage their time, and make sure the final product meets the needs of the people who will use it.

Here are three common methodologies:

## The Waterfall Model: A Step-by-Step Approach

Imagine building a waterfall, where water flows in one direction, from top to bottom, without going back up. The Waterfall Model works similarly: each step in the software development process must be finished before the next one can begin. It's a very traditional and ordered approach.

The typical steps in the Waterfall Model are:

1. **Requirements:** First, the team gathers all the detailed information about what the software needs to do. What features should it have? Who will use it?
2. **Design:** Based on these requirements, the team plans out how the software will be structured. This includes designing the user interface, databases, and how different parts of the program will work together.
3. **Implementation (Coding):** Developers write the actual code based on the design documents.

4. **Testing:** Once the coding is finished, the software is thoroughly tested to find and fix any bugs or problems.
5. **Deployment/Maintenance:** The software is released to users, and then ongoing support and updates are provided.

Think of it like this: You plan your entire trip, draw a detailed map, drive the whole way, check if you arrived safely, and then enjoy your destination.

**When it's good:** The Waterfall Model is often used for projects where the requirements are very clear and are not expected to change, like building a system for a specific, well-defined task. It's easy to understand and manage because of its strict order.

**Challenges:** If requirements change late in the process, it can be very difficult and expensive to go back and make changes, as you would have to "go back up the waterfall."

## The Spiral Model: Building in Cycles with Risk Management

The Spiral Model is like building software in a series of loops or spirals. Each loop through the spiral represents a phase of the project, and with each loop, the team creates a more complete and improved version of the software. A main focus of the Spiral Model is risk management – finding and dealing with possible problems early on.

Each "spiral" typically involves four main activities:

1. **Planning:** Setting goals, looking at alternatives, and defining limits for the current phase.
2. **Risk Analysis:** Identifying possible risks (like new technology not working, or requirements changing) and planning how to reduce them.
3. **Engineering:** Developing and testing a part of the software. This could be a prototype (a basic working model) or a more complete version.
4. **Evaluation:** Reviewing the results of the current phase and planning the next spiral.

Think of it like this: You plan a trip, consider risks like bad weather, try a short test drive (prototype), evaluate how it went, and then plan the next, longer part of the journey, repeating the risk assessment.

**When it's good:** The Spiral Model is great for large, complicated projects where requirements might not be fully clear at the start, or where there are many uncertainties. By building in cycles and constantly checking for risks, teams can adjust and make changes more easily than with Waterfall.

**Challenges:** It can be more complicated to manage than Waterfall, and if risks are not handled well, it can become very expensive.

## Agile Methodology: Flexible and Collaborative

Agile is a very popular and modern approach that focuses on being flexible, working together, and delivering working software quickly and often. Instead of one long, ordered process, Agile breaks down the project into small, manageable pieces called "sprints" (usually 1-4 weeks long).

Key ideas in Agile:

1. **Iterative and Incremental:** Software is built in small, repeating cycles, with new features added in each cycle.
2. **Customer Collaboration:** Users and customers are involved throughout the process, giving feedback regularly.
3. **Responding to Change:** Agile teams welcome changes, even late in development, because they are designed to be flexible.

4. **Working Software Over Documentation:** While documentation is important, the focus is on delivering working software that users can try out.
5. **Self-Organizing Teams:** Teams are allowed to decide how best to do their work.

Think of it like this: Instead of planning the whole trip, you plan a short segment, drive it, get feedback from passengers, and then plan the next segment, changing your route as needed.

**When it's good:** Agile is excellent for projects where requirements are likely to change, or where the final idea isn't perfectly clear from the start. It allows teams to adjust quickly and get feedback from users early and often. Many modern technology companies use Agile.

**Challenges:** It requires strong communication and dedication from everyone involved, and it might not be suitable for projects with very strict, unchanging rules.

## Different Stages in Development

Software development cycles involve distinct stages to solve problems. In the analysis phase, teams gather requirements, either fully upfront (like in Waterfall) or incrementally (Agile).

Design involves planning the software's structure; Waterfall creates a complete blueprint, while Spiral incorporates risk analysis.

During coding, Agile uses short "sprints" for continuous building and integration.

Testing, a key verification step, happens after all coding in Waterfall but continuously within Agile sprints.

Finally, maintenance involves ongoing support and updates, handled through structured releases in Waterfall or as continuous improvement in Agile and Spiral.

## Developing a Software Artifact

No matter which methodology a team chooses, the general process of developing a software artifact involves these main activities:

- **Requirements Gathering:** Understanding what the software needs to do.
- **Design:** Planning the software's structure and how it will work.
- **Coding (Implementation):** Writing the actual program code.
- **Testing:** Finding and fixing errors to make sure the software works correctly.
- **Deployment:** Releasing the software for users.
- **Maintenance:** Providing ongoing support, updates, and bug fixes.

Each methodology simply organizes and highlights these steps differently to best suit the project's needs and challenges.

## What Tools Do You Need?

Just like a carpenter needs tools, software developers use special tools to build programs. These help them write, organize, test, and manage code:

- **Integrated Development Environment (IDE):** A "one-stop shop" for programming, combining a code editor (for writing), a compiler/interpreter (to translate code for the computer), and a debugger (to find and fix errors).
- **Compiler/Interpreter:** Translates your code into a language the computer understands. A compiler translates the entire program at once, while an interpreter translates and runs it line by line.

- **Debugger:** A tool to help you find and fix errors (bugs) in your program by letting you pause the program and look at the code step-by-step.
- **Version Control System (VCS):** (like Git) Keeps track of every change to your code, allowing you to go back to older versions, compare changes, and work with others without writing over their work. It's like a history book for the project.
- **Text Editor:** A basic tool for writing and editing code, often with features like highlighting different parts of the code, even without all the features of an IDE.

These basic tools are essential for turning ideas into working software.

## Critical Thinking Questions

Imagine a team is building a new app for a local coffee shop. The coffee shop owner has a general idea of what they want, but they also want to be able to add new features and change things based on customer feedback. Which software development methodology (Waterfall, Spiral, or Agile) would be the best fit for this project, and why?

A government agency needs to develop a highly secure system for managing classified documents. The requirements for this system are extremely strict and cannot change once defined. Which methodology would likely be chosen, and what advantages would it offer in this specific scenario?

In the Spiral Model, "risk analysis" is a key part of each loop. Why is it so important for software development teams to constantly identify and deal with potential problems throughout a project, rather than just at the beginning?

Consider a very small software project, like a simple personal website for a school assignment. Which methodology would be the most efficient to use for this project, and why would the others be less suitable?

If a software development team is consistently missing deadlines using the Waterfall Model, what aspects of the other methodologies might they consider adopting to improve their project delivery?

How might the increasing popularity of artificial intelligence tools impact the way software development methodologies are applied in the future?

## Invitation: Build a Software Artifact!

Now that you've learned about different ways to build software, it's time to put your knowledge into practice! For your next project, work with your team to choose one of these software development methodologies (Waterfall, Spiral, or Agile) to guide your work.

### Here's your challenge:

Pick a simple software artifact to create. This could be a basic game, a simple calculator app, a small website, or even a detailed plan for an app you'd like to build someday.

Decide as a group which methodology makes the most sense for your project. Discuss the pros and cons of each in relation to your specific idea.

Follow the steps of your chosen methodology as you plan, design, create, and test your software artifact.

Document your process. Explain why you chose your methodology and how it helped (or challenged) your team during development.

This hands-on experience will give you a real taste of what it's like to develop software in a structured way!

## Questions (5)

**1. You are part of a team building a software project where every single step, like gathering all requirements and then doing all the design, must be completely finished before the next step can even start. Which model are you most likely using?**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. Agile Methodology
- B. Waterfall Model
- C. Spiral Model
- D. Iterative Model

**2. A software team is working on a large project with changing requirements, and they want to fix potential problems early. Which cyclical software development model is best suited for this?**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. Waterfall Model
- B. Spiral Model
- C. Agile Methodology
- D. Deployment Model

**3. A modern tech company needs to develop a new app quickly, and they want to get feedback from users constantly so they can make changes easily. Which software development methodology would be the best choice for them?**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. Waterfall Model
- B. Spiral Model
- C. Agile Methodology
- D. Maintenance Model

**4. In the Waterfall Model, what is the major challenge if the project's requirements change late in the development process?**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. It makes the software easier to test.
- B. It speeds up the deployment phase.
- C. It can be very difficult and expensive to go back and make changes.
- D. It encourages more collaboration.

**5. A team is using the Spiral Model. What is a key activity they do in each "spiral" or loop, which helps them deal with potential problems early?**

MULTIPLE CHOICE

**Choose the correct answer:**

- A. Final deployment of the software.
- B. Writing all the documentation.
- C. Risk Analysis.
- D. Completing all coding at once.

## Answer Keys & Solutions

### Questions

1. You are part of a team building a software project where every single step, like gathering all requirements and then doing all the design, must be completely finished before the next step can even start. Which model are you most likely using?

MULTIPLE CHOICE

**Correct Answer:**

- |                      |             |
|----------------------|-------------|
| A. Agile Methodology | ✗ Incorrect |
| B. Waterfall Model   | ✓ Correct   |
| C. Spiral Model      | ✗ Incorrect |
| D. Iterative Model   | ✗ Incorrect |

**Explanation:**

Think about the model that works strictly step-by-step, like water flowing downwards.

2. A software team is working on a large project with changing requirements, and they want to fix potential problems early. Which cyclical software development model is best suited for this?

MULTIPLE CHOICE

**Correct Answer:**

- |                      |             |
|----------------------|-------------|
| A. Waterfall Model   | ✗ Incorrect |
| B. Spiral Model      | ✓ Correct   |
| C. Agile Methodology | ✗ Incorrect |
| D. Deployment Model  | ✗ Incorrect |

**Explanation:**

Look for the model that uses repeating loops and emphasizes dealing with risks.

**3. A modern tech company needs to develop a new app quickly, and they want to get feedback from users constantly so they can make changes easily. Which software development methodology would be the best choice for them?**

MULTIPLE CHOICE

**Correct Answer:**

- |                      |             |
|----------------------|-------------|
| A. Waterfall Model   | ✗ Incorrect |
| B. Spiral Model      | ✗ Incorrect |
| C. Agile Methodology | ✓ Correct   |
| D. Maintenance Model | ✗ Incorrect |

**Explanation:**

Think about the methodology that is flexible and focuses on quick, repeating cycles with user input.

**4. In the Waterfall Model, what is the major challenge if the project's requirements change late in the development process?**

MULTIPLE CHOICE

**Correct Answer:**

- |  |             |
|--|-------------|
| A. It makes the software easier to test.                               | ✗ Incorrect |
| B. It speeds up the deployment phase.                                  | ✗ Incorrect |
| C. It can be very difficult and expensive to go back and make changes. | ✓ Correct   |
| D. It encourages more collaboration.                                   | ✗ Incorrect |

**Explanation:**

Consider the difficulty of going "back up the waterfall."

**5. A team is using the Spiral Model. What is a key activity they do in each "spiral" or loop, which helps them deal with potential problems early?**

MULTIPLE CHOICE

**Correct Answer:**

- |                                      |             |
|--------------------------------------|-------------|
| A. Final deployment of the software. | ✗ Incorrect |
|--------------------------------------|-------------|



B. Writing all the documentation.

✗ Incorrect

C. Risk Analysis.

✓ Correct

D. Completing all coding at once.

✗ Incorrect

**Explanation:**

Look for the activity that involves identifying and planning for potential issues.