

Software Development Methods

Textbook

Software Development Methods



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

Here are three common methodologies:

1. 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 completed before the next one begins. It's a very traditional, sequential approach.

The typical steps in the Waterfall Model are:

- **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?
- **Design:** Based on the 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.
- **Implementation (Coding):** Developers write the actual code based on the design documents.

- **Testing:** Once the coding is done, the software is thoroughly tested to find and fix any bugs or issues.
- **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 (requirements), draw a detailed map (design), drive the whole way (coding), check if you arrived safely (testing), and then enjoy your destination (deployment).

When it's good: The Waterfall Model is often used for projects where requirements are very clear and unlikely 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'd have to "go back up the waterfall."

2. 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 refined version of the software. A key focus of the Spiral Model is **risk management** – identifying and dealing with potential problems early on.

Each "spiral" typically involves four main activities:

- **Planning:** Defining goals, alternatives, and constraints for the current phase.
- **Risk Analysis:** Identifying potential risks (like new technology not working, or requirements changing) and planning how to reduce them.
- **Engineering:** Developing and testing a part of the software. This could be a prototype (a basic working model) or a more complete version.
- **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 leg of the journey, repeating the risk assessment.

When it's good: The Spiral Model is great for large, complex 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 adapt and make changes more easily than with Waterfall.

Challenges: It can be more complex to manage than Waterfall, and if risks aren't handled well, it can become very expensive.

3. Agile Methodology: Flexible and Collaborative

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

Key ideas in Agile:

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

- **Working Software Over Documentation:** While documentation is important, the focus is on delivering working software that users can try out.
- **Self-Organizing Teams:** Teams are empowered to decide how best to accomplish 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, adjusting your route as needed.

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

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

Developing a Software Artifact

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

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

Each methodology simply organizes and emphasizes 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 specialized 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 whole 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 execution and examine code step-by-step.
- **Version Control System (VCS):** (like Git) Tracks every change to your code, allowing you to revert to older versions, compare changes, and collaborate with others without overwriting work. It's a project's history book.
- **Text Editor:** A basic tool for writing and editing code, often with features like syntax highlighting, even without all the features of an IDE.

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

Critical Thinking Questions:

1. 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?
2. 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?
3. 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?

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.