Waterfall Model

Scenario Story:
You are building a payroll system for a local bank. The requirements are clear, stable, and unlikely to change. The project is short, and the team has ample expertise.
Why Use Waterfall?

- **Clear, Stable Requirements:** The model works best when all requirements are known upfront and unlikely to change[1][2][3].
- **Simple to Implement:** Easy to manage and track progress[3].
- **Good for Short Projects:** Works well for small to medium-sized projects with fixed scope[1][2].

Advantages:

- **Simple and easy to implement.**
- **Clear milestones and deliverables at each stage.**
- **Good for projects where quality is more important than cost or schedule[3].**
- **Easy project management and tracking[3].**

Disadvantages:

- **No working software until late in the cycle[2][3].**
- **Difficult to accommodate changes once the process has started[1][2].**
- **High risk if requirements are not well understood[2][3].**
- **Not suitable for complex, long-term, or object-oriented projects[1][2].**

V-Model

Scenario Story:
You are developing a safety-critical railway signaling system. Each feature must be rigorously tested as soon as it is developed. Requirements are detailed and signed off by regulators.
Why Use V-Model?

- **Disciplined Testing:** Each development phase has a corresponding testing phase[4].
- **High Quality:** Ensures quality is built in from the start[4].
- **Regulated Environment:** Ideal for safety-critical systems[4].

Advantages:

- **Simple and easy to use[4].**
- **Early defect detection through rigorous testing[4].**
- **High discipline and quality assurance[4].**

Disadvantages:

- **High risk and unpredictable for complex projects[4].**
- **Difficult to go back to previous stages after testing[4].**
- **Not suitable for long or ongoing projects[4].**

Incremental Model

Scenario Story:
You are developing a new email client for a startup. Basic features are clear, but more advanced features will be added in increments as user feedback is received.
Why Use Incremental?

- **Flexibility:** Features can be added in stages5.
- **Early Delivery:** Basic functionality is delivered quickly5.
- **Customer Feedback:** Users can provide input for future increments5.

Advantages:

- **Early and quick delivery of functional software5.**
- **Flexible to changes and new requirements5.**
- **Easier testing and debugging in smaller increments5.**
- **Customer feedback can be incorporated after each increment5.**

Disadvantages:

- **Requires good upfront planning and design5.**
- **Can become uncoordinated if not well managed5.**
- **Potential for rework if requirements change significantly5.**
- **Integration challenges as new increments are added5.**

Spiral Model

Scenario Story:
You are building a new online payment gateway for a fintech company. Requirements are unclear, and there are high risks in security and compliance. You need to manage risks and prototype solutions.
Why Use Spiral?

- **Risk Management:** Each iteration includes risk assessment6.
- **Prototyping:** Allows for building and testing prototypes6.
- **Flexibility:** Adapts as you learn more about the project6.

Advantages:

- **Strong focus on risk management6.**
- **Flexible and adaptable to changing requirements6.**
- **Customer involvement and feedback throughout6.**
- **Early error detection and continuous refinement6.**

Disadvantages:

- **Costly and time-consuming due to continuous risk analysis and prototyping6.**
- **Complex to manage, especially for small teams6.**
- **Risk of inadequate documentation6.**

Concurrent Model

Scenario Story:
You are developing a mobile app with a web backend. The frontend and backend teams need to work in parallel to meet a tight deadline.
Why Use Concurrent?

- **Parallel Development:** Different teams work on different components at the same time[7].
- **Faster Delivery:** Speeds up development by overlapping activities[7].
- **Integration Focus:** Encourages early integration and problem detection[7].

Advantages:

- **Allows for parallel development of components[7].**
- **Reduces overall development time[7].**
- **Early integration and problem detection[7].**

Disadvantages:

- **Complex to manage due to overlapping activities[7].**
- **Potential integration challenges[7].**
- **Requires good communication and coordination[7].**

Agile Models (Scrum, XP, Kanban, ASD, DSDM)

Scenario Story:
You are part of a startup building a social media platform. The market is changing fast, and users want new features every week. The team is small, creative, and adaptable.
Why Use Agile?

- **Adaptability:** Responds quickly to changing requirements[8][9].
- **Frequent Delivery:** Working software is delivered in short cycles[8][9].
- **Customer Collaboration:** Customers are involved throughout the project[8][9].

Advantages:

- **Frequent delivery of working software[8][9].**
- **Welcomes changing requirements even late in development[8][9].**
- **Promotes collaboration and customer involvement[8][9].**
- **Continuous improvement and feedback[8][9].**

Disadvantages:

- **Requires high customer involvement[9].**
- **Can be difficult to measure progress[9].**
- **Risk of scope creep and technical debt[9].**
- **Documentation can be lacking[9].**
- **Can be less predictable in terms of time and cost[9].**

Unified Process (UP)

Scenario Story:
You are managing a large enterprise software project with many stakeholders and complex requirements. The project will be developed in phases, with each phase delivering a set of features.
Why Use UP?

- **Architecture-Centric:** Focuses on building a robust architecture[10].
- **Iterative and Incremental:** Delivers features in iterations[10].
- **Use-Case Driven:** Requirements are captured as use cases[10].

Advantages:

- **Strong focus on architecture and scalability[10].**
- **Iterative and incremental delivery[10].**
- **Good for managing complex, long-term projects[10].**

Disadvantages:

- **Complex and requires strong management[10].**
- **Can be time-consuming due to extensive documentation and planning[10].**
- **Not suitable for small projects or teams[10].**

Summary Table

Model	**Scenario Story Example**	**Advantages**	**Disadvantages**
Waterfall	Bank payroll system upgrade	Simple, clear milestones, easy management[1,2,3]	No working software until late, inflexible, high risk if requirements change[1,2,3]
V-Model	Railway signaling system	Early defect detection, high discipline, easy to use[4]	High risk, hard to backtrack, not for complex/long projects[4]
Incremental	Startup email client	Early delivery, flexible, customer feedback, risk management[5]	Needs good planning, can be uncoordinated, integration challenges[5]
Spiral	Fintech payment gateway	Risk management, flexible, customer involvement, early error detection[6]	Costly, time-consuming, complex to manage, documentation risk[6]
Concurrent	Mobile app with web backend	Parallel work, faster delivery, early integration[7]	Complex management, integration challenges[7]
Agile	Social media platform	Frequent delivery, adaptable, customer collaboration, continuous feedback[8,9]	High customer involvement, scope cree less predictable, documentation risk[9]
Unified Proc.	Enterprise software with many stakeholders	Robust architecture, iterative delivery, good for complex projects[10]	Complex, time-consuming, not for sma teams[10]