1. **Waterfall**

Features: Linear, sequential approach where each phase (requirements, design, coding, testing, deployment) must be completed before moving to the next. It requires well-defined requirements and is less flexible.

Example: Developing a military defence system that needs precise documentation and has rigid processes, with no room for mid-project changes. (government)

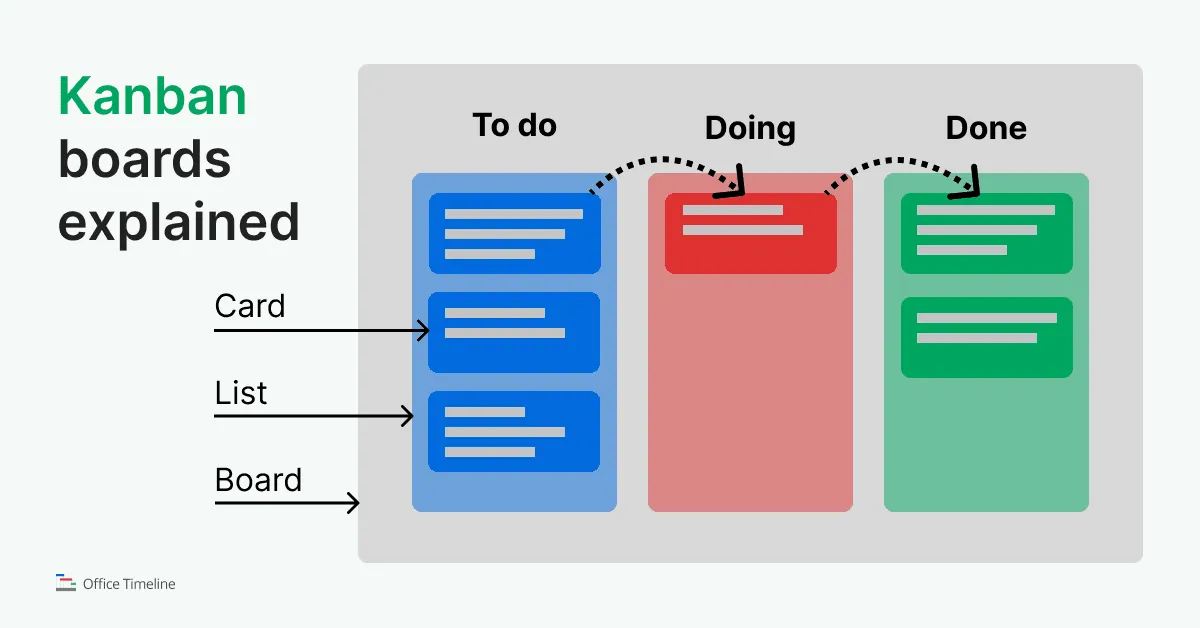
1. **Agile**

Features: Iterative and flexible, Agile focuses on delivering work in small increments called sprints, allowing regular feedback and adaptation.

Example: Developing a fitness tracking app, where user needs change frequently, requiring ongoing adjustments and iterative releases.

1. **Kanban**

Features: Visualizes tasks on a board, limiting work in progress to optimize workflow and increase efficiency. It focuses on continuous delivery without time-boxed sprints.



Example: Managing a software development help desk, where support tickets are moved through stages such as "Open," "In Progress," and "Closed" on a Kanban board.

1. **Scrum**

Features: A subset of Agile, Scrum works in fixed-length sprints (usually 2-4 weeks). It includes roles like Scrum Master and Product Owner, and ceremonies such as daily standups and sprint reviews.

Example: Building an online booking system for travel, with each sprint focusing on specific functionalities such as payment gateways or search features.

1. **Lean**

Features: Focuses on reducing **waste** and maximizing value. Lean emphasizes efficiency, continuous improvement, and delivering only what the customer needs.

Example: Launching a startup MVP for an app, where the goal is to develop only essential features and avoid unnecessary functionality to speed up time-to-market.

1. **DevOps**

Features: Emphasizes collaboration between development and IT operations teams. It focuses on automation, continuous integration, and delivery (CI/CD) for faster, more reliable releases.

Example: For an online food delivery app, the DevOps methodology allows the team to release new features (e.g., payment integration, order tracking) rapidly. Developers push code changes to a shared repository, where automated tests ensure everything works as expected. Once passed, the code is automatically deployed to a staging environment for further validation. With continuous monitoring in place, any performance bottlenecks in production are quickly identified and resolved, ensuring high uptime and a smooth user experience.

1. **Extreme Programming (XP)**

Features: Stresses technical excellence and customer satisfaction. Practices include pair programming, test-driven development (TDD), and frequent releases.

Example: Developing high-quality software for a financial institution, where pair programming and constant refactoring ensure code stability and security.

1. **Spiral Model**

Features: Combines iterative and Waterfall approaches, focusing on risk analysis. Each phase (planning, risk assessment, development) is revisited in a spiral for refinement.

Example: Developing a complex medical device software, where risks are identified and addressed in each iteration to ensure safety and compliance.

1. **Feature-Driven Development (FDD)**

Features: Agile methodology focused on delivering features in short iterations. It emphasizes detailed planning and design with feature-centric development.

Example: Building a large-scale retail website, where teams break down the project into features (e.g., product catalog, checkout process) and work on each feature in detail.

1. **Rapid Application Development (RAD)**

Features: Prioritizes rapid prototyping and user feedback over strict planning. It's ideal for projects with flexible requirements and short development cycles.

Example: Developing a custom CRM tool for a business, where prototypes are built quickly and adjusted based on continuous feedback from users.