Advanced Software Engineering Assignment 1

Supervised by professor Ahmed El-mahlawy.

Student name : ماهر جمعه اسماعيل سيد

Academic number : 152300311

الفرقة الثالثة لائحة حديثة

Feature	**Waterfall**	**Agile**	**DevOps**
Definition	A linear, sequential development process where each phase must be completed before moving to the next.	An iterative and incremental approach that focuses on adaptability and customer collaboration.	A cultural shift that integrates development and operations to enable continuous delivery and automation.
Approach	Rigid, phase-based execution	Iterative, feedback-driven	Continuous integration, delivery, and deployment
Phases	Requirements → Design → Implementation → Testing → Deployment → Maintenance	Iterations (Sprints) with ongoing feedback and delivery	Continuous development, automated testing, deployment, and monitoring
Flexibility	Limited; changes are difficult and costly	Highly adaptable, supports evolving requirements	Constant improvements based on feedback and automation
Risk Management	High risk due to late-stage testing and feedback	Lower risk as testing is integrated into each sprint	Minimal risk with automation ensuring early issue detection
Speed of Delivery	Slow, due to extensive upfront planning and sequential execution	Faster, as development occurs in incremental cycles	Extremely fast due to automated workflows and continuous deployment
Customer Involvement	Low, as feedback is gathered primarily at the end	High, with regular stakeholder collaboration	Continuous collaboration between developers, operations, and customers
Testing	Conducted after full development	Integrated into each development cycle	Automated testing embedded within CI/CD pipelines
Team Structure	Hierarchical with predefined roles	Cross-functional, self-organizing teams	Cross-functional teams including development, operations, and security
Cost Efficiency	Can be costly due to late-stage error detection and changes	More cost-effective due to early defect detection and flexibility	Optimized through automation and proactive monitoring
Best Suited For	Large-scale projects with well-defined requirements and minimal expected changes	Projects with evolving requirements and a need for rapid adaptation	Cloud-based applications, microservices, and high-frequency deployments

Waterfall Methodology

1. **Requirements Gathering** – Comprehensive documentation of project requirements before development begins.
2. **System Design** – Creation of architecture, system specifications, and technical design.
3. **Implementation** – Development of the software based on the predefined requirements.
4. **Testing** – Validation and verification of the completed system.
5. **Deployment** – Delivery of the final product to the customer.
6. **Maintenance** – Ongoing support and updates.

Pros:

- Well-documented and structured process.
- Ideal for projects with stable requirements.
- Easier to manage in regulated industries.

Cons:

- High risk if requirements change mid-development.
- Late-stage testing can result in costly fixes.
- Slower delivery compared to iterative methods.

Agile Methodology

1. **Iterative Development** – Continuous releases in small increments.
2. **Collaboration** – Close engagement with stakeholders for feedback.
3. **Adaptive Planning** – Ability to respond to changes quickly.
4. **Customer-Centric Approach** – Ensuring the final product aligns with user needs.

Pros:

- High flexibility and adaptability to changes.
- Frequent feedback ensures customer satisfaction.
- Encourages collaboration and innovation.

Cons:

- Requires strong team coordination and discipline.
- Less structured documentation compared to Waterfall.
- Can lead to scope creep if not well managed.

DevOps Methodology

1. **Continuous Integration (CI)** – Frequent code integration to detect issues early.
2. **Continuous Delivery (CD)** – Rapid and automated deployment of software.
3. **Infrastructure as Code (IaC)** – Automating infrastructure management.
4. **Monitoring & Feedback** – Constant tracking of performance and security.

Pros:

- Accelerates development and release cycles.
- Enhances collaboration between teams.
- Ensures high software quality and reliability.

Cons:

Requires cultural transformation and investment in automation tools.

Complex implementation for traditional organizations.

Security concerns with continuous deployment