Online Event Management Syst

Requirement Specification for the Project.

1. Objective(s):

The Software Development Life Cycle (SDLC) aims to produce high-quality software that meets or exceeds customer expectations while completing the project within time and cost estimates.

For the Online Event Management System, the objectives are:

- Facilitate online event creation, registration, and management.
- Provide a seamless user experience for event attendees and organizers.
- Ensure secure payment processing and ticketing.
- Enable reporting and analytics for event organizers.

2. Problem Analysis

Organizing events such as seminars, workshops, conferences, and cultural programs is often a challenging task. In many cases, event planning and management are done manually, which creates several problems:

I. Time-Consuming Process
   - Registration, ticket booking, and scheduling are handled through paper forms or emails.
   - This slows down the process and makes it difficult to manage a large number of participants.

II. Data Management Issues
   - Event details, participant records, and payments are scattered across different systems (spreadsheets, documents, emails).

   - This increases the chance of errors, duplication, and data loss.

III. Limited Accessibility
   - Without an online system, participants must visit physically or rely on organizers for updates.
   - Important announcements may not reach everyone on time.

IV. Lack of Transparency
   - Payment tracking, seat availability, and registration confirmation are not always clear.
   - This can create trust issues between participants and organizers.

V. Inefficient Communication
   - Manual methods make it difficult to send reminders, event updates, or changes to all participants quickly.

VI. Scalability Problems
   - As the number of events and participants increases, manual systems cannot handle the workload effectively.

3. Methodology

The Software Development Life Cycle (SDLC) is a structured process followed for the development of software projects. It provides a framework for planning, designing, developing, testing, deploying, and maintaining software in a systematic way. By following SDLC, the quality of software and the overall development process can be improved.

Stage 1: Planning and Requirement Analysis

- **Performed by:** Senior team members
- **Inputs from:** Customers, sales department, market surveys, and domain experts
- **Main Activities:**

    - Conducting requirement analysis (fundamental stage in SDLC)
    - Planning the basic project approach
    - Performing feasibility studies (economic, operational, and technical)
    - Planning for quality assurance requirements
    - Identifying project risks
- **Outcome:** Well-defined technical approaches to implement the project with minimum risks.

Stage 2: Defining Requirements

- **Purpose:** To clearly define and document all product requirements.
- **Approval By:** Customers and market analysts
- **Main Activity:** Creation of the **Software Requirement Specification (SRS)** document, which contains all the functional and non-functional requirements to be designed and developed during the project.

Stage 3: Designing the Product Architecture

- **Input:** Software Requirement Specification (SRS)
- **Output:** Design Document Specification (DDS)
- **Activities:**
    - Proposing multiple design approaches for the product architecture
    - Reviewing DDS with stakeholders
    - Selecting the best design approach based on risk, robustness, modularity, budget, and time
- **Defines:** Product module architecture, communication and data flow, and internal design of modules.

Stage 4: Building or Developing the Product

- **Input:** Design Document Specification (DDS)

- **Activity:** Actual coding and software development, where developers follow coding standards and best practices.
- **Tools:** Compilers, interpreters, and debuggers
- **Languages:** Depending on the project, languages such as C, C++, Java, or PHP may be used.

Stage 5: Testing the Product

- **Testing Integration:** Though testing is performed in all stages, this phase focuses on dedicated testing.
- **Activities:**
  - Defect reporting and tracking
  - Bug fixing and retesting
- **Goal:** Ensure the product meets the quality standards defined in the SRS and functions as expected.

Stage 6: Deployment and Maintenance

- **Deployment:** The product is formally released into the market. In some cases, a staged release is performed.
- **Activities:**
  - Conducting User Acceptance Testing (UAT) with a limited group of end-users
  - Adjusting the system based on feedback before the final release
- **Maintenance:** Continuous support and updates are provided to ensure smooth operation and accommodate changes over time.

4. SDLC Models

Different models of the Software Development Life Cycle (SDLC) provide alternative approaches to software development. The choice of model depends on the nature, size, and complexity of the project. The commonly used models are:

4.1 Waterfall Model

- Process: Linear and sequential; each phase must be completed before the next begins.
- Characteristics:
  - No overlapping phases
  - Progress flows downward like a waterfall
- Use Cases:
  - Requirements are very clear, fixed, and well-known
  - Suitable for small and simple projects

4.2 V-Model (Verification & Validation Model)

- Process: Testing is planned in parallel with development.
- Shape: Development phases on one side of "V", testing phases on the other, with coding linking them.
- Focus: Early verification and validation to ensure higher quality.
- Use Cases:
  - Small projects
  - Stable and clearly defined requirements

4.3 Iterative Model

- Process: Software is built in small parts and enhanced through repeated iterations.
- Characteristics:
  - Each version improves with added features
  - Design changes are possible at each iteration
- Use Cases:
  - Requirements are clear and understandable
  - Large applications
  - Future changes are expected

4.4 Spiral Model

- Process: Software is developed in iterative cycles (spirals), each containing four phases:
  - Identification
  - Design
  - Build
  - Evaluation and risk analysis
- Characteristics: Strong focus on risk management and cost evaluation.
- Use Cases:
  - Large and complex projects
  - Projects needing frequent releases
  - When prototypes are required
  - Risk and cost evaluation is critical

4.5 Prototyping Model

- Process: A working prototype is built to gather user feedback, which helps refine the final product.
- Characteristics:
  - Prototype = small-scale version of the final system
  - Helps clarify requirements at an early stage
- Use Cases:
  - When requirements are not clear initially
  - When customer involvement and feedback are crucial

4.6 Agile Model

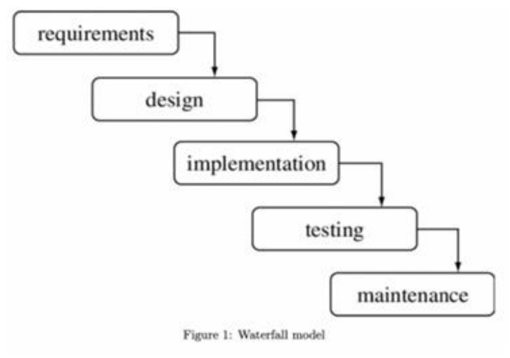
- Process: Iterative and incremental development with high flexibility.
- Characteristics:
  - Work divided into small iterations (sprints)
  - Scope of each iteration is pre-defined
  - Encourages adaptability and customer collaboration
- Use Cases:

    - Projects with frequently changing requirements
    - When fast delivery is needed
    - Projects focused on customer feedback

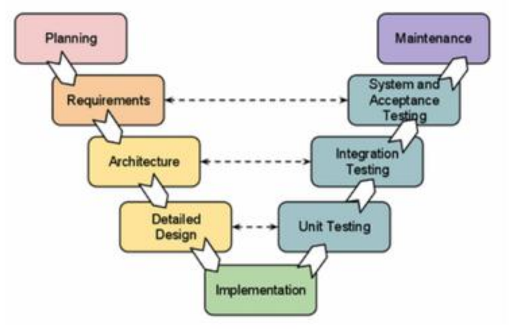
4.7 Model Selection for Online Event Management System (OEMS) – Agile Model

For the Online Event Management System, the Agile Model is highly suitable because:

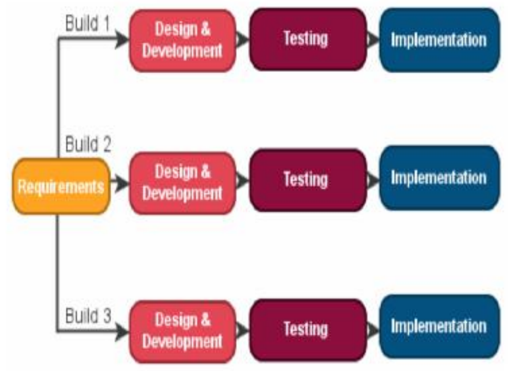
- The system may need frequent updates or new features (e.g., organizers want custom event categories, dynamic pricing, different payment gateways, or new notification methods). Agile allows continuous improvements.
- Agile works in small sprints, so functional parts like event creation, user registration, ticket booking, and payment integration can be delivered quickly and improved through feedback.
- Customer collaboration is a core of Agile. Since event organizers and users may suggest changes after testing the system, Agile makes it easy to adjust requirements.
- Agile reduces risks by providing early working software that stakeholders can test and approve before final deployment.
- In an online platform like OEMS, user experience and adaptability are critical, and Agile ensures faster delivery with flexibility.



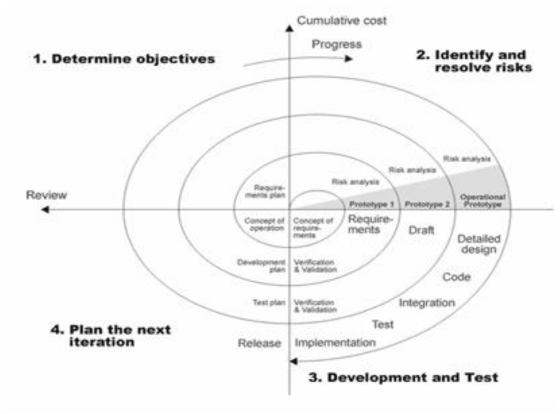
Waterfall Model



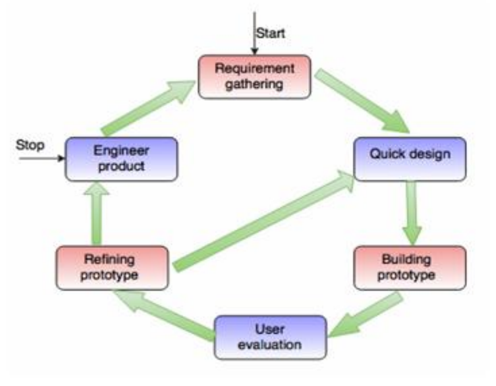
V Model



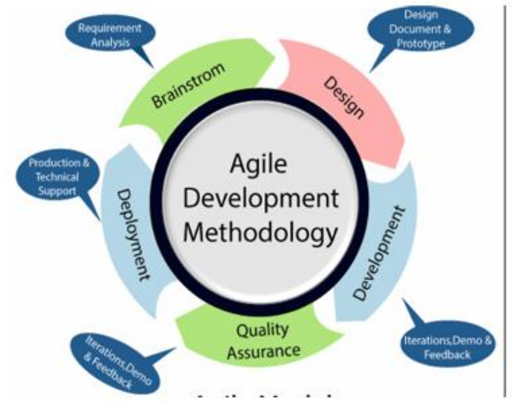
Iterative Model



Spiral Model



Prototype Model


Requirement Analysis
Design Document & Prototype
Brainstrom
Design
Production & Technical Support
Agile Development Methodology
Deployment
Development
Quality Assurance
Iterations,Demo & Feedback
Iterations,Demo & Feedback

Agile Model

Model	**Process**	**Advantages**	**Use Cases**
Waterfall	Linear, sequential; next phase starts only after the previous is completed	- Simple and easy to manage - Clear milestones - Works well for small projects	- Requirements are fixed, clear, and stable - Small/simple projects
Iterative	Build initial version, then improve through repeated iterations	- Early working software - Flexible for changes - Easier debugging and testing	- Large applications - Requirements are clear but may evolve
V-Model	Development phases on one side of "V", testing phases on the other; coding links them	- Testing happens in parallel with development - High quality & reliability	- Small projects - Requirements are clear and stable
Spiral	Cyclic process with 4 phases: Identification, Design, Build, Evaluation (with risk analysis)	- Strong risk management - Suitable for large & complex projects - Frequent releases possible	- Large/complex projects - Risk and cost evaluation required - Prototyping needed
Agile	Iterative & incremental; development in	- Very flexible - Fast delivery of working software - Customer	- Projects with changing requirements -

Model	Process	Advantages	Use Cases
	small sprints with customer involvement	collaboration ensures satisfaction	When speed and adaptability are important
Prototyping	Build a prototype to gather feedback before final system	- Helps clarify requirements early - User involvement ensures correctness - Reduces risk of misunderstandings	- Requirements unclear at start - Customer feedback is essential

Example: Implementation in Online Event Management System

Here, a table is created for the SDLC Model for an Online Event Management System (OEMS). We selected categories that are suitable for this project, such as changing requirements, user involvement & feedback, fast delivery, risk analysis, efficiency & scalability, and security & dependability.

We assigned a priority to each category based on the project's requirements. Then we analyzed which SDLC model supports which category. By combining the priorities with the support provided by each model, we calculated a total score for each SDLC model.

The scores are:

- Waterfall: 6
- V-Model: 9
- Iterative: 20
- Spiral: 23
- Agile: 26

- Prototype: 18

From the analysis, we see that the most important categories for an OEMS—changing requirements, user feedback, and fast delivery—are best supported by the Agile model.

Table 1: Comparison matrix with different models

Priority	**Criteria**	**Waterfall**	**V-Model**	**Iterative**	**Spiral**	**Agile**	**Prototype**
5	**Changing requirements**	**No**	**No**	**Yes**	**Yes**	**Yes**	**Yes**
4	**User involvement & feedback**	**No**	**Limited**	**Yes**	**Yes**	**Yes**	**Yes**
5	**Fast delivery / Incremental updates**	**No**	**No**	**Yes**	**Yes**	**Yes**	**Partial**
3	**Risk analysis**	**No**	**No**	**No**	**Yes**	**Limited**	**No**
4	**Efficiency & scalability**	**No**	**Yes**	**Yes**	**Yes**	**Yes**	**Limited**
3	**Security & dependability**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Limited**
Total (out of 30)	**Over all**	**6**	**9**	**20**	**23**	**26**	**18**