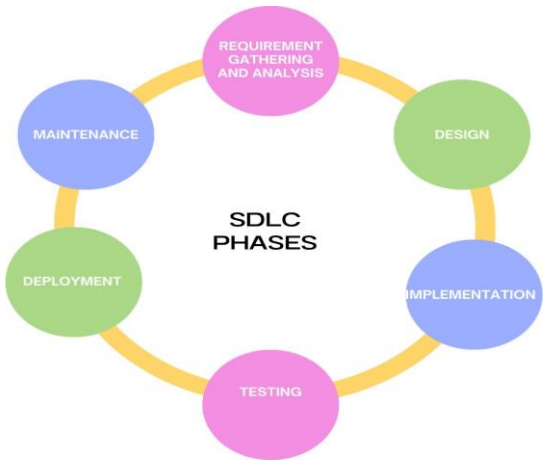
DAY- 2

Assignment 1: SDLC Overview - Create a one-page infographic that outlines the SDLC phases (Requirements, Design, Implementation, Testing, Deployment), highlighting the importance of each phase and how they interconnect.



1. Requirement Gathering and Analysis: This is the first phase where the project team interacts with the stakeholders to understand their requirements. They gather all the relevant information about what the software should do and define the system requirements.

2. Design: In this phase, the system and software design documents are prepared according to the requirement specification document. This helps in specifying hardware and system requirements and also helps in defining the overall system architecture.

3. Implementation or Coding: The design documents are used in this phase as inputs and the actual coding is done here. It's the phase where the developers start writing code according to the requirements and the design discussed.

4. Testing: After the code is developed, it's tested against the requirements to make sure the product is actually solving the needs addressed and gathered during the requirements phase.

5. Deployment: Once the software is tested and ready to be deployed, it's made available to the users. Sometimes it may also involve installing the software on the client side.

6. Maintenance: After the deployment, any modifications, changes, and enhancements that are required in the operational software are taken care of during this phase.

Assignment 2: Develop a case study analyzing the implementation of SDLC phases in a real-world engineering project. Evaluate how Requirement Gathering, Design, Implementation, Testing, Deployment, and Maintenance contribute to project outcomes.

- Requirement Gathering and Analysis:
  - Aim: Develop a non-invasive system for emotion recognition.
  - Stakeholders: Human-computer interaction, therapy, advanced robotics.
  - Emotions to recognize: Valence, Arousal, Dominance.
- Design:
  - Use of DREAMER Database.
  - Acquisition of ECG and EEG signals
  - Extraction of features using ML classifiers.
- Implementation or Coding:
  - Development of machine learning models.
  - Fusion with EEG, ECG signals.
  - Development of a Non - invasive model.
- Testing:
  - Use of video clips to trigger different emotions in human subjects.
  - Testing DREAMER dataset
  - Testing of multifarious algorithms.
- Deployment:
  - Use in human-computer interaction interfaces, therapy, and advanced robotics.
- Maintenance:
  - Regular updates and enhancements based on user feedback and advancements in the field.

Conclusion

The case study demonstrated how a clear understanding of requirements, a well-thought-out design, careful implementation, rigorous testing, smooth deployment, and diligent maintenance can lead to the successful realization of a complex project like emotion recognition.

Assignment 3: Research and compare SDLC models suitable for engineering projects. Present findings on Waterfall, Agile, Spiral, and V-Model approaches, emphasizing their advantages, disadvantages, and applicability in different engineering contexts.

WATERFALL MODEL

The main concept of this model is that only when one development level is completed will the next one be initiated.

ADVANTAGES

- The Waterfall method is also well known by software developers
- The Waterfall Model works well for smaller projects where requirements are very well understood
- Cost-effectiveness: Time spent early in the software production cycle can lead to the greater economy at later stages.

DISADVANTAGES

- The test is seen as a “final inspection”, an analogy to a manufacturing inspection before handing over the product to the customer.
- Inflexible
- Poor model for complex and object-oriented projects.

APPLICABILITY

Ideal for projects with fixed scope and requirements, and no changes are expected.

V MODEL

The V-Model is an extension of the Waterfall model that emphasizes the importance of testing and validation. It involves testing each phase of the development process in parallel with its corresponding phase, forming a V-shaped lifecycle.

ADVANTAGES

- Ensures thorough testing and validation at each phase.
- The objectives of testing are changing, and specific for each test level

DISADVANTAGES

- This model is applicable mostly to big companies because the model needs a lot of resources.
- The amount and the intensity of the test levels should be tailored according to the specific needs of the project.

APPLICABILITY

Applicable for projects where requirements are well-defined and there's a clear understanding of the system architecture.

SPIRAL MODEL

The Spiral model combines the iterative nature of prototyping with the systematic aspects of the Waterfall model. It consists of multiple cycles, each including four phases: planning, risk analysis, engineering, and evaluation.

ADVANTAGES

- The Spiral Life Cycle Model is a very flexible model.
- Good for large and mission-critical projects.
- Allows for early identification and mitigation of risks.

DISADVANTAGES

- Doesn't work well for smaller projects.
- Evaluating the risks involved in the project can shoot up the cost and it may be higher than the cost of building the system.
- Risk analysis requires highly specific expertise.

APPLICABILITY

Suitable for large-scale projects with high risk and evolving requirements, emphasizing risk management and iterative development.

AGILE MODEL

Agile is an iterative and incremental approach to software development, emphasizing flexibility, collaboration, and customer feedback. It breaks the project into small iterations or sprints, with each iteration delivering a working increment of the software.

ADVANTAGES

- Customer satisfaction is rapid, continuous development and delivery of useful software.
- Product is developed fast and frequently delivered.

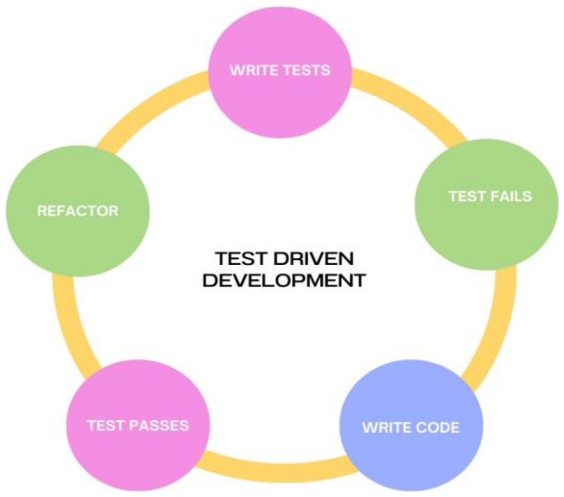
DISADVANTAGES

- It is not useful for small development projects.
- It requires an expert project member to take crucial decisions in the meeting.
- Cost of Agile development methodology is slightly more as compared to other development methodology.

APPLICABILITY

Ideal for projects requiring flexibility and collaboration, delivering incremental value through iterative development and continuous feedback loops.

Assignment 4: Create an infographic illustrating the Test-Driven Development (TDD) process. Highlight steps like writing tests before code, benefits such as bug reduction, and how it fosters software reliability.



Test Driven Development is the process in which test cases are written before the code that validates those cases. It depends on repetition of a very short development cycle. Test driven Development is a technique in which automated Unit tests are used to drive the design and free decoupling of dependencies.

The following sequence of steps is generally followed:

- Add a test – Write a test case that describes the function completely. In order to make the test cases the developer must understand the features and requirements using user stories and use cases.
- Run all the test cases and make sure that the new test case fails.
- Write the code that passes the test case.
- Run the test cases.
- Refactor code – This is done to remove duplication of code.
- Repeat the above mentioned steps again and again.

BENEFITS:

- **Bug Reduction**: TDD leads to fewer bugs in the codebase since tests are written to verify the expected behavior of the code. This helps catch and prevent defects early in the development process.
- **Improved Code Quality**: With tests driving the development process, the resulting code tends to be cleaner, more modular, and easier to maintain. TDD encourages developers to write code that is focused on fulfilling specific requirements.
- **Fostering software reliability:** TDD fosters software reliability by ensuring that the code behaves as intended. Each test represents a specific behavior or requirement that the code must fulfill.
- **Faster Feedback Loop**: TDD provides rapid feedback on the correctness of the code. When a test fails, developers know immediately that something is wrong and can address it before moving on to other tasks.

Assignment 5: Produce a comparative infographic of TDD, BDD, and FDD methodologies. Illustrate their unique approaches, benefits, and suitability for different software development contexts. Use visuals to enhance understanding.

<u>Test Driven Development (TDD):</u>

- Focus: TDD emphasizes the implementation of a feature within a software application or product.
- Process:

1. Add a test case that fails because the specified functionality doesn’t exist.
2. Update the code to make the test case pass.
3. Result: The feature is implemented in the system.

- Main Participants: Developers.
- Starting Point: Test cases.
- Focus Area: Unit tests.
- Language Used: Similar to the one used for feature development (e.g., programming language).
- Collaboration: Required only between developers.
- Approach : Development Practice

Behavior Driven Development (BDD):

- Focus: BDD focuses more on a software application's behavior.
- Process:
    1. Write the behavior of the application (usually in simple English language).
    2. Write automated scripts.
    3. Implement the functional code.
    4. Check if the behavior is successful; if not, fix it.
    5. Repeat for another behavior.
- Main Participants: Developers, Customers, QAs.
- Starting Point: Scenarios.
- Focus Area: System requirements.
- Language Used: Simple English language for writing behavior/scenarios.
- Collaboration: Required between all stakeholders.
- Approach: Team methodology.

Feature Driven Development (FDD):

- Focus: FDD is about planning and tracking progress.
- Process: It involves breaking down features into smaller, manageable tasks and tracking their completion.
- Main Participants: Typically project managers and development teams.
- Starting Point: Features.
- Focus Area: Feature planning and progress tracking.
- Language Used: N/A (more about planning and management).
- Collaboration: N/A (primarily management-driven).
- Approach: Project management practice.

--END--