**Assignment 2: 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.**

**Comparison Table**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Methodology** | **Approach** | **Process** | **Benefits** | **Suitable Contexts** |
| **TDD** | Write tests before writing code. | 1. Write Test → 2. Run Test → 3. Write Code → 4. Run Tests → 5. Refactor → 6. Repeat | - Reduces bugs early - High test coverage - Improves code quality | - High reliability projects - Complex systems - CI/CD environments |
| **BDD** | Focus on behavior from a user perspective. | 1. Define Behavior → 2. Convert to Specs → 3. Write Code → 4. Run Tests → 5. Refactor → 6. Repeat | - Improves collaboration - Ensures user requirements - Clear documentation | - Agile environments - Stakeholder interaction - Clear communication |
| **FDD** | Develop features that deliver business value. | 1. Develop Model → 2. Build Feature List → 3. Plan by Feature → 4. Design by Feature → 5. Build by Feature | - Aligns with business objectives - Tangible progress - Efficient project management | - Large-scale projects - Business value focus - Clear milestones |

**TDD (Test-Driven Development):**

* Write tests before writing code.
* Focus on testing individual units of code.

**Process**:

1. Write a Test
2. Run the Test (it should fail)
3. Write Code to pass the Test
4. Run Tests again (they should pass)
5. Refactor Code
6. Repeat

**Advantages:**

**1.**It improves the code quality.

2.It reduces the bugs

**Disadvantages**:

1.Time consuming process.

2.Developing test cases of every scenario is difficult.

**Benefits**:

* Reduces bugs in early stages.
* Ensures high test coverage.
* Improves code quality and maintainability.

**Suitable Contexts**:

* Projects requiring high reliability.
* Complex systems with numerous unit tests.
* Continuous Integration/Continuous Deployment (CI/CD) environments.

**Visual**: Circular or linear flow chart illustrating the TDD cycle.

**2. BDD (Behavior-Driven Development):-**

* Focus on the behavior of the application from the user's perspective.
* Uses natural language descriptions for test cases.

THE Three main principles of BDD are:

1. Focus on the behavior or outcome.
2. Collaboration between teams.
3. Use a common language for communication and understanding.

**Process**:

1. Define behavior in plain English (Given, When, Then)
2. Convert behavior into executable specifications.
3. Write code to meet the specifications.
4. Run all behavior tests.
5. Refactor Code
6. Repeat

**Advantages**:

1.Encourages communication and collaboration between developers, testers, and Business analysts.

2.Translates requirements into plain language, making them understandable to everyone involved.

3.Focuses on the behavior of the system from a user’s perspective, ensuring the software meets business objectives.

**Disadvantages**:

1.Detailed tests can slow down initial development.

2.Success relies on active, consistent collaboration.

**Benefits**:

* Improves collaboration between developers, QA, and non-technical stakeholders.
* Ensures the application meets user requirements.
* Provides clear documentation.

Suitable Contexts:

* Agile development environments.
* Projects with significant stakeholder interaction.

**3.FDD (Feature-Driven Development):-**

* Focus on developing features that deliver business value.
* Uses a feature-centric approach to development.

**Process**:

1. Develop an Overall Model
2. Build a Feature List
3. Plan by Feature
4. Design by Feature
5. Build by Feature

**Benefits**:

* Ensures alignment with business objectives.
* Delivers tangible progress with each feature.
* Facilitates efficient project management.

**Advantages**:

1. Clear picture of project scope.
2. User-cetric approach
3. Decreased need of meetings
4. Works well for long projects.

**Disadvantages:**

1. Dependence on lead developers.
2. No written documentation to the client

**Suitable Contexts**:

* Large-scale projects with clear feature requirements.
* Organizations focusing on delivering business value.
* Teams require clear milestones and progress tracking.