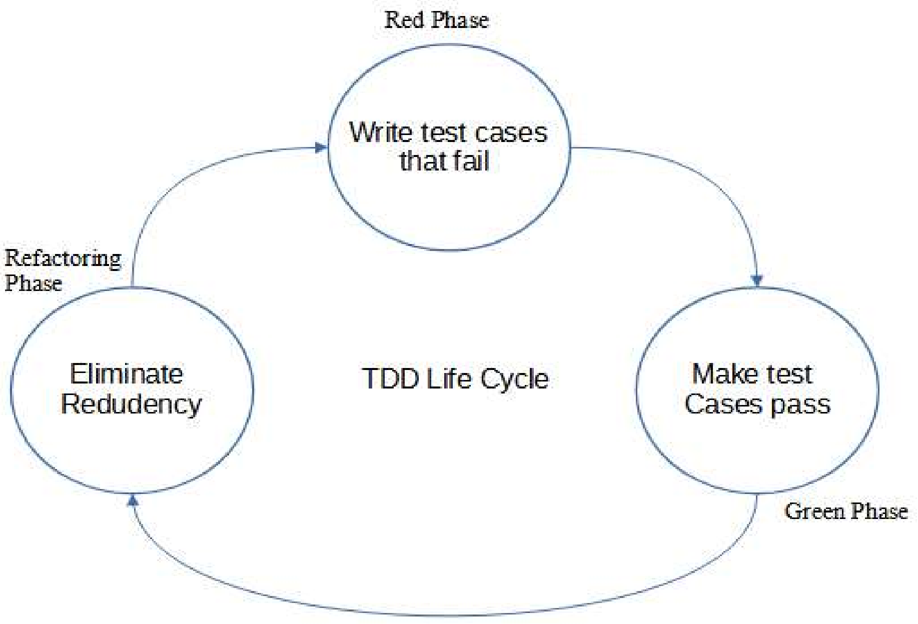
ASSIGNMENT 2 13-05

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

Test driven development (TDD) is an iterative methodology that prioritizes the creation of checking against test cases at every stage of software development, by converting each component of the application into a test case before it is built and then testing and tracking the component repeatedly.

TDD Life Cycle:



Approach: TDD involves writing tests before writing the corresponding code. TDD integrates seamlessly with CI/CD pipelines, allowing for automated testing with each code commit. Developers follow the Red-Green-Refactor cycle, where they start with a failing test (Red), make it pass with minimal code changes (Green), and then refactor the code to improve its design without changing its behavior.

Benefits:

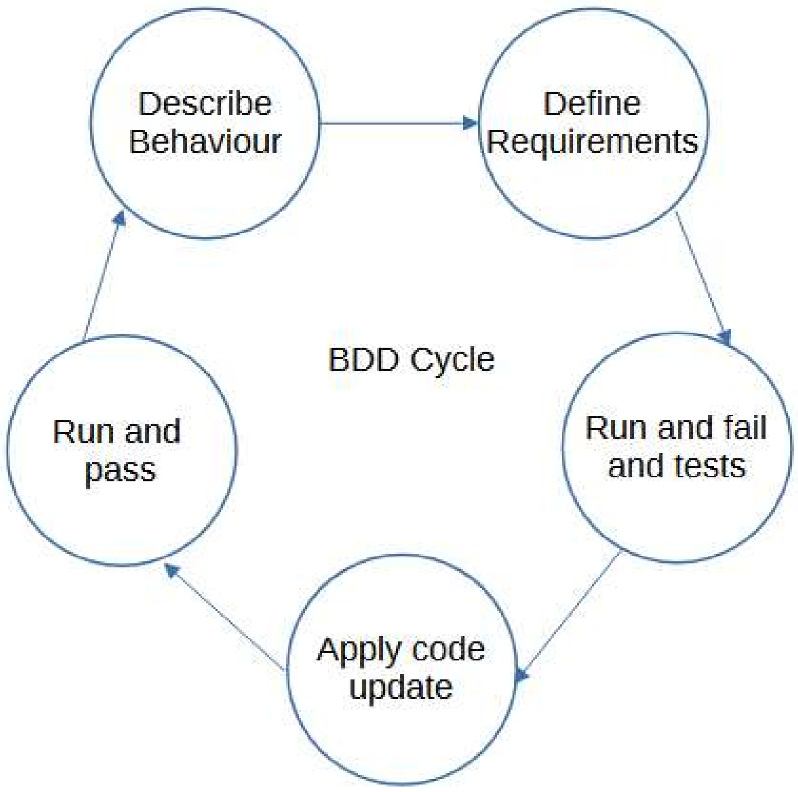
- Regression Testing: TDD ensures that existing functionality remains intact when new features are added or existing code is modified.
- Improved Code Quality: TDD encourages writing modular, well-structured code by focusing on writing tests that verify the desired behavior.
- Code Documentation: Test cases serve as executable documentation, providing insights into the expected behavior of the code.
- Agile Development: TDD aligns well with Agile principles, enabling rapid iteration and adaptation based on changing requirements.
- Refactoring: TDD supports refactoring the process of restructuring code without changing its external behavior. This encourages continuous improvement and maintenance of code quality without sacrificing reliability.

Suitability for Different Saftware Developments:

- **Web Development:** TDD ensures the reliability of web applications by thoroughly testing both frontend and backend components.
- **Embedded Systems:** TDD helps ensure the reliability and robustness of embedded systems by testing individual components and functionalities.
- **API Development:** TDD can be particularly useful in API development to ensure that APIs behave as expected and provide the desired functionality.

Behavior Driven Development (BDD) is a software development methodology that emerged from the principles of Test-Driven Development (TDD) and Agile software development practices. BDD focuses on collaboration among developers, testers, and business stakeholders to ensure that software development aligns with business goals and user expectations.

BDD Life Cycle:



Approach: BDD extends TDD by focusing on the behavior of the system from the end-user's perspective. BDD often starts with user stories, which describe user needs and system behavior from the perspective of stakeholders. BDD defines acceptance criteria for each user story, outlining the conditions that must be met for the feature to be considered complete.

Benefits:

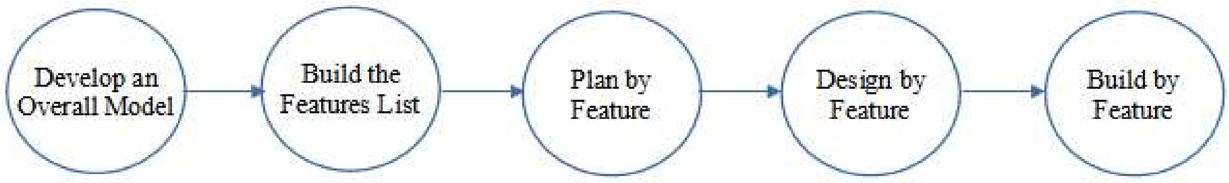
- Reduced Rework: BDD ensures that development efforts are focused on delivering features that meet user needs, reducing the likelihood of rework due to misunderstood requirements.
- Improved Collaboration: BDD encourages collaboration between developers, testers, and business stakeholders by providing a shared understanding of the system's behavior.
- Stakeholder Engagement: BDD encourages active participation from stakeholders throughout the development process, leading to better alignment between technical teams and business objectives.
- Automated Acceptance Testing: BDD scenarios can be automated to serve as acceptance tests, providing a reliable means of verifying system behavior.

Suitability for Different Saftware Developments:

- **Enterprise Software:** BDD facilitates communication and collaboration between technical teams and business stakeholders in enterprise software development.
- **Mobile App Development:** BDD ensures that mobile apps meet user expectations and business requirements by focusing on behavior from the user's perspective.
- **API Development:** BDD can help ensure that APIs provide the desired functionality and meet user needs by describing their behavior in terms of scenarios and examples.

Feature Driven Development (FDD) is an iterative and incremental software development methodology that focuses on delivering tangible, working software features in short iterations.
FDD places a strong emphasis on the features of the software being developed and employs a structured, model-driven approach to software development.

FDD Life Cycle:



Approach: FDD starts with creating a comprehensive feature list based on the project requirements and stakeholder inputs. This feature list serves as the backbone of the development process, guiding the team's efforts throughout the project lifecycle. FDD divides the development process into short, time-boxed iterations, with each iteration focusing on delivering one or more features.

Benefits:

- FDD enables web development teams to prioritize and deliver user-facing features incrementally, leading to faster time-to-market and improved user experience.
- FDD provides a systematic approach to developing embedded systems by breaking down the development effort into manageable features and iterations.
- FDD allows for continuous feedback and refinement, ensuring that the web application meets user expectations and requirements.
- FDD enables product development teams to prioritize features based on customer needs and market demand.

Suitability for Different Saftware Developments:

- **Game Development:** FDD helps manage the complexity of game development by prioritizing and developing features incrementally, ensuring that each feature is well-designed and thoroughly tested.
- **IoT Applications:** FDD facilitates the development of interconnected features and functionalities in IoT applications by identifying and developing features that provide value to end-users.