Research Week 12

1. What is the difference between TDD and BDD?

Test-Driven-Development (TDD) is a methodology where a test that satisfies a requirement of the application is written first, then minimal code is written to make the test pass. The test is then refactored to make it fail, thus requiring refactoring of the application code to make the test pass again. This is known as the Red-Green-Refactor cycle, based on the JUnit test result bar being red when a test fails, and green when a test passes. TDD is good for testing the functional implementation of an application in parts. This generally makes for more concise code written by developers.

Behavior-Driven-Development (BDD) is a methodology where a desired behavior of the application is written first, usually in plain English, with all people involved in the project contributing, then test code is written based on the behavior described in the first part of BDD. The functional application code to satisfy the behavior is written, then rewritten if the code doesn't work properly. This makes for more results-based code written by the entire team. BDD is more result-based than TDD and can test configuration and SQL statements.

In summary, TDD involves writing a test first, then writing code to implement a requirement of the application to make the test pass. BDD involves defining the desired behavior, then writing a test based on that behavior, then writing the application code to meet the requirements of the behavior. BDD also requires fewer unit tests than TDD.

References

TDD vs. BDD: Key Differences. (2020, April 27). Retrieved February 11, 2023 from https://phoenixnap.com/blog/tdd-vs-bdd

TDD Vs BDD – Analyze The Differences With Examples. (2023, January 12). Retrieved February 11, 2023 from https://www.softwaretestinghelp.com/tdd-vs-bdd/

Difference between BDD vs TDD in Software Engineering. (2022, May 27). Retrieved February 11, 2023 from https://www.geeksforgeeks.org/difference-between-bdd-vs-tdd-in-software-engineering/

TDD vs BDD: Choosing The Suitable Framework. (2022, March 11). Retrieved February 11, 2023 from https://www.lambdatest.com/blog/tdd-vs-bdd/

Research Week 12

2. What does mocking a class allow you to do?

When writing code, sometimes we need to test the methods of a class without relying on other classes and/or services. This is usually not possible in normal testing, so we create a fake, or mock, class/object that supplies the needed external parameters for the method under test allowing us to verify the method's proper behavior. Mocking a class allows us to remove the external dependencies of a class and substitute for them with a mock object.

References

What is Mocking in Testing? (2017, October 21). Retrieved February 11, 2023 from https://piraveenaparalogarajah.medium.com/what-is-mocking-in-testing-d4b0f2dbe20a

Stubbing and Mocking with Mockito and JUnit. (2022, August 30). Retrieved February 11, 2023 from https://semaphoreci.com/community/tutorials/stubbing-and-mocking-with-mockito-2-and-junit

The Concept of Mocking (2008, August 16). Retrieved February 11, 2023 from https://dzone.com/articles/the-concept-mocking