Differences Test Driven Development, Feature Driven Development, Behavior Driven Development

Test Driven Development (TDD) is a software development methodology that emphasizes on writing automated tests before writing the actual code. The process involves writing a test case, running the test case (which will fail as the code is yet to be written), and then writing the code to pass the test case. TDD is a developer-centric approach, and the focus is on ensuring that the code meets the requirements specified in the tests.

Feature Driven Development (FDD) is an agile software development methodology that focuses on delivering features incrementally. FDD is based on the five-step process of developing an overall model, building a feature list, planning by feature, designing by feature, and building by feature. FDD is focused on delivering the feature set and ensures that the overall process is efficient and effective.

Behavior Driven Development (BDD) is a software development methodology that focuses on the behavior of the system from the perspective of the stakeholders. BDD is based on the principles of Test Driven Development (TDD) but places more emphasis on the behavior of the system. BDD uses a natural language syntax to describe the system's behavior and ensures that the behavior of the system is well defined and understood by all stakeholders.

In summary, TDD focuses on ensuring that the code meets the requirements specified in the tests, FDD focuses on delivering features incrementally, and BDD focuses on the behavior of the system from the perspective of stakeholders.

Summary of advantages of moving from Centralized Source Code (CVCS) to Distributed Version Control (DVCS)

- DVCS has the biggest advantage in that it allows you to work offline and gives flexibility. You have the entire history of the code in your own hard drive, so all the changes you will be making in your own server or to your own repository which doesn’t require an internet connection, but this is not in the case of CVCS.

- DVCS is faster than CVCS because you don’t need to communicate with the remote server for each and every command. You do everything locally which gives you the benefit to work faster than CVCS.
- Working on branches is easy in DVCS. Every developer has an entire history of the code in DVCS, so developers can share their changes before merging all the ‘sets of changes to the remote server. In CVCS it’s difficult and time-consuming to work on branches because it requires to communicate with the server directly.
- Merge conflicts with other developer’s code are less in DVCS. Because every developer work on their own piece of code. Merge conflicts are more in CVCS in comparison to DVCS.
- In DVCS, sometimes developers take the advantage of having the entire history of the code and they may work for too long in isolation which is not a good thing. This is not in the case of CVCS