A discussion on TDD and BDD

As a typical confusion these days between the new emerging developments that take place are test driven and behavior driven. Today, the entire discussion is to understand how both the developments work and what is the difference in force that drives the development model. Along with this we will also try to evaluate with a small case study.

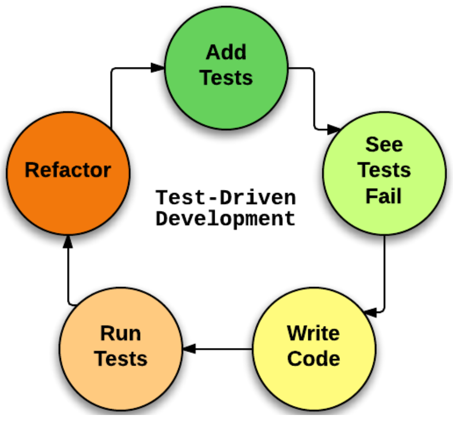
To start with is a definition to understand TDD and BDD:

- **Test-driven development** (**TDD**) is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests, only. This is opposed to software development that allows software to be added that is not proven to meet requirements.

- **Behavior-driven development** (**BDD**) is a software development process that emerged from test-driven development (TDD). Behavior-driven development combines the general techniques and principles of TDD with ideas from domain-driven design and object-oriented analysis and design to provide software development and management teams with shared tools and a shared process to collaborate on software development.

We will first discuss some features of Test Driven Development. i.e.

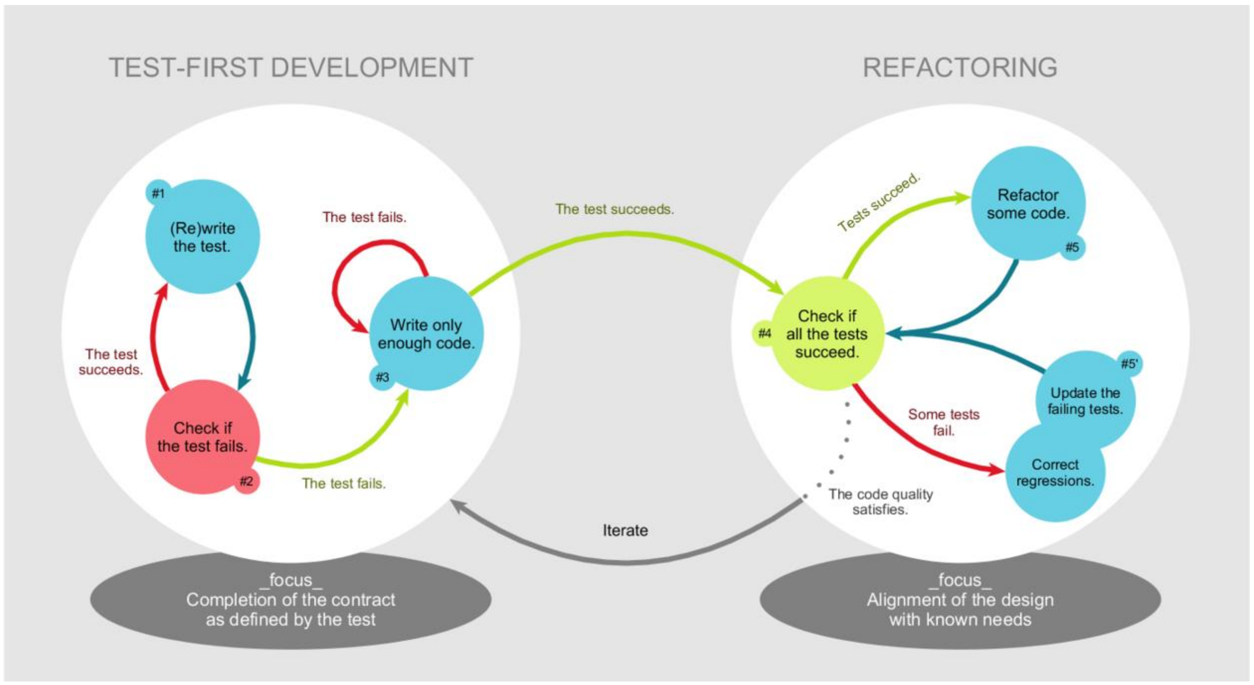
1. A process in which software is developed; code is written and parallel tested for defects to assure quality of semantics and logics used.
2. It allows having very high test coverage.
3. Higher test coverage assures of defect free software.
4. It also reduces the chances to have bugs during testing, which can otherwise be difficult to track down.



Steps followed by a TDD process

This makes it easier to conclude to the following points of the Test Driven Development as:

- TDD promotes writing more Test Cases which in turn, programmers tend to be more productive and a better quality of code is achieved.
- TDD also created an environment in which the programmers rarely needed the need to invoke a debugger. With this if an unexpected error arrives, it was easier to use version controlling to restore code to last functional point, instead of invoking a debugger.
- TDD helps correct designing of a program along with validation.
- TDD allows developers to concentrate on small and individual units as primary job is to pass the tests. This allows greater level of confidence on the code.
- TDD limits the number of defects due to large number of defects and eliminating defects early avoids lengthy and tedious debugging in later stages of the project.
- TDD leads to smaller, more focused classes, looser coupling, and cleaner interfaces. It also contributes to the modularized code to replace mock test units with real versions for production as it allows units to be tested independently and integrated together later.
- TDD allows covering each path of the code via automated tests for unit testing.



But still, in spite of so many benefits TDD still failed as it is Expensive and it slows down and delays project launch. Changes in project requirements means old tests are no longer usable. This also means having lot more work than just testing the outcomes and testing extensively also tedious and boring and is costly to conduct. Because of these reasons many reputed developers and BAs do not use TDD at all. Even if TDD is used then the developers face lots of questions like when and what to be tested and if the specifications are met and will this code deliver business value?

These days we actually analyze and understand what processes will make designing software with a difference. The upcoming points will actually emphasize on the Business rather than testing:

- Individuals and interactions over testing, processes and tools.
- Working software over comprehensive test documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a test plan.

Thus looking at an overall perspective of using Business Driven Development, we see the following benefits i.e.

1. With BDD, all the involved parties have a strong understanding of the project and they can all have a role in communication and actually have constructive discussions. BDD increases and improves collaboration.
2. Everyone gets strong visibility into the project's progression.
3. BDD puts great importance on the business value and needs. By setting priorities with the client, based on the value it provides, developers are able to provide a better result because they have a strong understanding of how the client thinks.
4. It reduces misconceptions and misunderstanding and makes it easier for new members to join the working process.
5. With BDD, focus is on the behavior, which has a stronger impact than the implementation itself.
6. Teams using BDD are in general much more confident that they won't break the code, and have better predictability when it comes to their work.
7. By improving the quality of the code, you are basically reducing costs of maintenance and minimizing the project's risks.

On a larger understanding, we can come to a common conclusion for the comparative study:

- ➢ It makes it very clear that having BDD is better driven than TDD.
- ➢ It saves cost and time and is highly flexible in requirement phase.
- ➢ Saves efforts and ensures delivery on time.

Thus Behavior driven Development derives examples of different expected behaviors of the system. It enables writing the examples in a language using the business domain terms to ensure easy understanding by everyone involved in the development including the customers. Gets the examples ratified with customer from time to time by means of conversations.

Focuses on the customer requirements (examples) throughout the development and uses examples as acceptance tests.

Let us get more accustomed to the TDD – BDD concept via a Case Study.

- ❖ Consider a developer on a team responsible for the company accounting system. One day, a business person asks to implement a reminder system to remind clients of their pending invoices.
- ❖ Because he is practicing BDD per this case study; (versus TDD), he sits down with that business person and start defining behaviors.
- ❖ Thus specifications created by a Business User:
    1. It “adds a reminder date when an invoice is created”
    2. It “sends an email to the invoice's account's primary contact after the reminder date has passed”
    3. It “marks that the user has read the email in the invoice”
- ▪ Approach taken by a Behavior Driven Development will be:
    - ▪ What's the default reminder date going to be?
    - ▪ How many days before the invoice due date?
    - ▪ Are those business days or just calendar days?
    - ▪ What happens if there's not a primary contact associated with the account?
    - ➢ Thus, in this way, Behavior-Driven Development is a tool to aid
        - ▪ Collaboration
        - ▪ Start a conversation between the two departments
        - ▪ It’s also a way to clarify the scope of a desired feature
        - ▪ And get better estimates from the dev team
- Approach taken by a Test Driven Development will be:
    - ▪ It "after_create an Invoice sets a reminder date to be creation + 20 business days"
    - ▪ It "Account#primary_payment_contact returns the current payment contact or the client project manager"
    - ▪ It "InvoiceChecker#mailer finds invoices that are overdue and sends the email"
    - ➢ In this way, Test-Driven Development is a tool to aid
        - ▪ These tests are helpful, but only helpful to one group of people: engineers.
        - ▪ Better Validation of functionalities instead of Verification if Features required in the application.

Summary of the User Story and its Conclusion:

- ✓ This focus on behavior during development makes the test useful as verification that developer is building the right feature, not just that the code is correct.

- ✓ The mistaken business person will either get distracted by the details or take this new knowledge and try to micromanage things that the developer knows more about (proper database design, code reuse).
- ✓ In this case study we saw that writing a one-line overview of a specific behavior will bore the business person. They'll view it as a poor use of their time or grow anxious to describe the next behavior while it's on their mind.
- ✓ This can be assisted by use of BDD instead of TDD during development time and instead use test to validate and verify the developed product in later stages.