BDD - Assignment
Group 6

Dušana Milinković (s194741),
Emma Andreea Chirlomez (s194780),
Paulo Lima (s194729), Dario Cannistra (s194830),
Renjue Sun(s181294) and Diana Podoroghin (s194768)

02160 Agile Object-oriented Software Development
Danmarks Tekniske Universitet
March 19, 2021

1 Introduction

BDD is the abbreviation of 'Behavior Driven Development'. BDD emerged from TDD (the abbreviation of 'Test Driven Development') and is an extension of it. TDD is a development practice while BDD is a team methodology [1]. TDD is more about ensuring programmers deliver high quality code and BDD has the edge over TDD in communication and feedback [4]. From the agile software team's perspective, we need to lay more emphasis on BDD for it can drive us to understand each other's work, continuously receive feedback within the team and also to brainwash ideas about what features and values we need to bring to our product. Figure 1 illustrates the relationship between BDD and TDD. We agreed that when we work individually on a divided task, we focus on the discipline of TDD to pass all predefined test for a satisfactory output. When working as a team, we integrate all individual outputs and to see how they can contribute to different features of our software. This follows the concept of BDD.

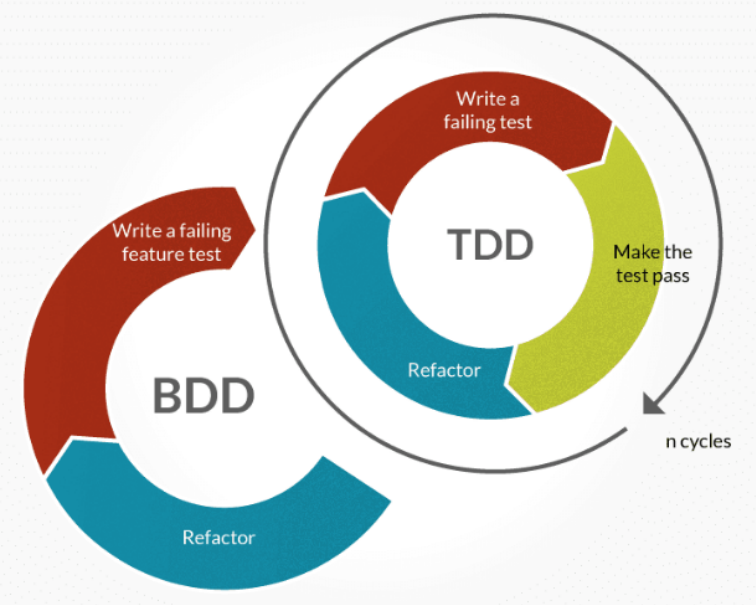

Figure 1: The relationship between BDD and TDD [4]

In this assignment, we managed to go through the process of BDD and investigated why and how should we be agile.

2 User story

User stories are very crucial to work in an agile way. User stories are written in natural language and they are simple and short description that aim to explain what a user would like to achieve.

To check if each story does what is supposed to do we defined acceptance criteria. We structured each stories and acceptance criteria with the *Given When, Then* BDD notation. This format will be very useful to convert them into automated test by using tools like Cucumber in the future.

In order to check the User Stories, you can click on our GitHub repository and go to the Projects section named *Development*.

3 Definition of done

In order for a User Story to be marked as done, we established some steps that we must follow.

The definition of done comprises:

- The story implementation meets all the acceptance criteria
- Every acceptance criteria have at least a test case associated
- Unit tests are passed and coverage threshold is reached
- Code is well documented
- Code refactoring is completed
- The issue related to the user story is closed, by merging the branch to origin.

4 Why we are Agile

In order to have a well-organized and productive work on the project, our team needs to be agile. To do so we agreed to pursue the following concepts. First of all, we decided to divide all the work in small tasks because in this way we will be able to integrate the future feedback faster and also to minimize the time used to determine and fix problems. Moreover, it will help to have a good time management as it is easier to estimate the time needed for a certain task. Secondly, we have created several chats on different platforms such as Teams, GitHub, etc. to be able to easily communicate and review on regular basis of our performance and outcomes, help each other when someone is stuck as well as include discussions on improving and moving forward. If it will happen that something is not working out(someone is not working together with the team) we will try to discuss that between our team members and try to find a proper solution for that certain problem. For instance, if the person cannot handle the task, other team members will always help and if that is not a solution we can try to find another task that the person will be able to do better.

4.1 Agile methods

4.1.1 Version control

Version control software is used to keep track of, manage, and secure changes to file [2]. A version control system records all the changes made to a file or set of files, so a specific version may be called later if needed. The version control system ensures that all workstations (6 group members in our case) are always working on the latest version of the repository. We decided to use version control for it can benefit us from managing and protecting the source code, keeping track of all modifications made to the code, comparing different versions of the code and supporting our group's workflow. As a rule of thumb, GitHub is probably the standard for open source products [3] and our group members have had some experience with it, so we decided to use Git for our version control. Figure 2 describes the version control process of our group. Everyone should have a working copy of our GitHub repository on his/her own device. When starting working on the project, a group member should update changes from the repository first and commit his/her update to the repository when the work is done.

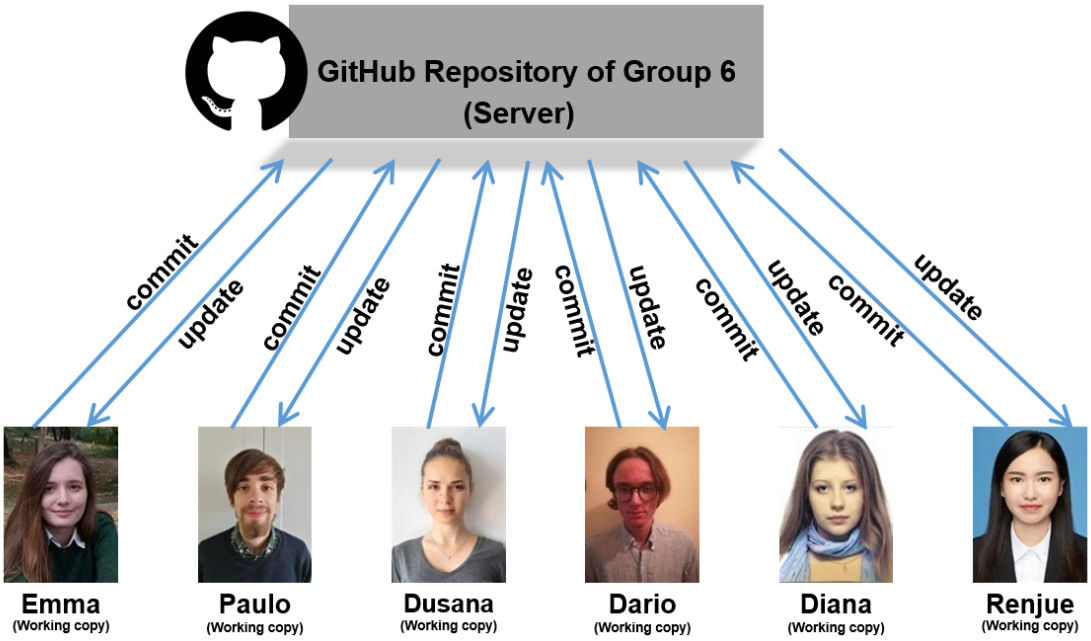

Figure 2: Version control of Group 6

4.1.2 Code refactoring

Refactoring code to make it more maintainable and extendable has become a more mainstream practice [5]. Although the external behavior of the code will not be changed by refactoring, code refactoring can increase the readability and extensibility, improve the maintainability and decrease the complexity of the code. These are of great importance in large group projects, because every group member should take the responsibility to read other's code and then extend it and the software system is more likely to collapse with the increase of its complexity. We decided to consistently do code refactoring with our existing code and each group member should be responsible for it.

References

[1] David Adsit. Tdd vs bdd. URL https://www.pluralsight.com/blog/software-development/tdd-vs-bdd.

[2] Davis Adam L. *Modern Programming Made Easy*, chapter Version Control. Apress, 2020.

[3] Zandstra Matt. *Php Objects, Patterns, and Practice*, chapter Version Control with Git. Apress, 2016.

[4] Jithin Nair. Tdd vs bdd - what's the difference between tdd and bdd. URL https://blog.testlodge.com/tdd-vs-bdd/.

[5] Joseph Yoder. Refactoring at the core of agile software development. In *Proceedings of the 10th International Conference on Aspect-oriented Software Development Companion, Aosd.11*, pages 51–52. ASSOC COMPUTING MACHINERY, 2011.