BDD in DevOps

Agnes Petäjävaara
agnespet@kth.se
April, 2021

I. Introduction

In teams with bad teamwork, there is often a me-first mentality with a lack of trust and transparency. People's unique strengths and weaknesses are not valued, the goal is not in focus, and conflicts exist. Teams that have malfunctioning teamwork rarely deliver on time or meet the expectations set by stakeholders. (1) Bad teamwork is nothing that automagically disappears once you leave school. To be able to change a negative atmosphere you actively have to work on it. A common way is to follow different strategies and this is where Agile came in about 20 years ago. Agile is an umbrella term for multiple different methodologies that all focus on collaboration, customer feedback, and small rapid releases. All to make the best out of your specific team, catch the individual strengths and weaknesses, and ensure that you deliver on time and with high quality. One of the core values of Agile is that each team should follow the practice that suits the team and their goals the best, improves weaknesses and avoids threats. (2; 3; 4)

As you probably understand from the title of this essay, we will not focus on the Agile world, but on the DevOps culture. DevOps stands for the collaboration between Development and Operations, and works towards optimizing the flow of value from idea to end user. It is not a replacement of agile but rather a successor that works great in parallel with its predecessor. This essay focuses on one of these methodologies from the Agile umbrella, Behavior-driven development (BDD), and how and why it can be adapted to the DevOps culture.

II. Behavior-driven development

Behavior-driven development (BDD) focuses on the behaviors and requirements of users and stakeholders. The goal is to contribute to the direct outcomes of a project by encouraging multiple stakeholders to better collaborate. This as a way to minimize communication gaps and ensure that both technical and non-technical speakers have a shared understanding of the project. BDD has the goal to remove unnecessary pitfalls and to make the development process more efficient and the overall cost lower.

BDD comes from the agile practice TDD (Test-Driven Development). Following TDD means that you write your tests first and then write the code needed to make them pass. Following this practice is very popular in the agile world since generates a high code coverage, minimizes the risk of missed bugs, and avoids spending time on writing unnecessary code. A problem with TDD is that it goes from the inside out. Developers tend to focus too much on their units and lose track of the bigger goal, meaning that a team can deliver high-quality code but that ends up being nothing like what the stakeholder asked for. A total waste of time! This is where BDD comes in. BDD goes from the outside-in, where the behavior of the project deliverable is more important than the exact implementation details of the tech behind it. (5) Figure 1 visualizes the differences.

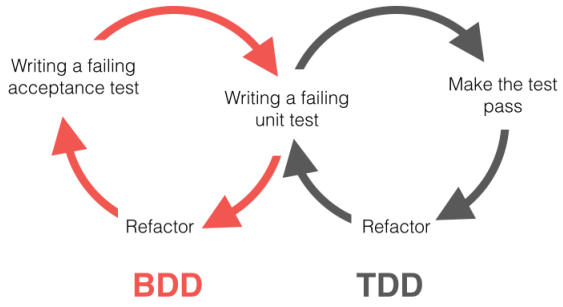

Fig. 1. BDD vs TDD Simplified Life Cycle (6)

III. Getting up to speed with BDD

BDD is primarily a conceptual approach meaning that no particular tools or programming languages are needed. This makes it an inclusive tool that breaks down complex requirements and reformulate them so that everyone have an equal understanding of the expectations of the project. The practice is divided into three main steps; Discovery, Formulation, and Automation (see figure 2) Discovery is the first step and aims to overcome blind spots and to reduce the bottleneck of learning. The discovery phase focuses on gaining a shared understanding of the challenges ahead. This is also the phase where all the different behaviors of the project are found and where the acceptance criteria (which should be fulfilled at the end of the project) are written. When the behaviour of a project has been discovered and expectations agreed on, it is time to formulate how it can later be automated. Since BDD is primarily an Agile practice, there is no given BDD syntax to use when working with the formulation. But one commonly used syntax is Gherkin's "Given, When, Then" (7). More about Gherkin and other tools and frameworks in section VII.

After the behaviors have been discovered and implementation details have been formulated, it is time to automate them. The core idea when deciding on which behaviour to implement is to always be guided by their priority; you should ask yourself "What is the next most important thing the system does not yet do?" and start implementing it. (9)

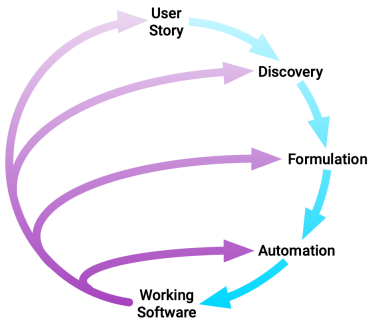

Fig. 2. The different phases of BDD (8)

IV. Three Amigos

An important rule of BDD is that you find agreement with your team and stakeholders on all three steps of the implementation process. “The three amigos” is an Agile concept referring to the relationship and collaboration between business, development, and testing (10). The idea behind it is to make people holding this different expertise collaborate when defining what to do and agree on the ’definition of done’(11). This to ensure that the problem is attacked from all different perspectives before, during, and after the implementation. In the Discovery phase, it is fundamental to have people with expertise from business, development, and quality, present when decisions are made to avoid miscommunications. The ’Business amigo’ should have the perspective of the cases that provide business value, the ’Development amigo’ should have the perspective of technical constraints to consider, and the ’Quality amigo’ should have the perspective of the user expectations and potential edge-cases to be considered. The term “amigo” should not be synonymized with an individual. ”Amigo” should rather be seen as ”expertise”. The same individual can have multiple expertise, therefore it can be more or fewer than three individuals present when the “Three Amigos” workshop takes place. (12)

V. The DevOps lifeloop [1]

DevOps emerged from the Agile world similarly as BDD emerged from TDD. Both DevOps and Agile focus on development, testing, and deployment, but the problem faced in the traditional Agile world is that it excludes the IT teams from the active work making them deal with the end project and the end of the project life cycle. While agile focuses on the flow of software from ideation to code completion, DevOps extends this focus to also include delivery and maintenance. While agile strongly advocates collaboration between development teams and their stakeholders, DevOps goes above by also valuing the collaboration with the operations team. When agile is about being responsive to change, DevOps is all about delivering faster by automating as much as possible of the assembly line. (4)

Yes, the DevOps culture emerged from the Agile world, but it does not mean it will replace it. They work great side by side, just like BDD and TDD. DevOps six phases are; planning, building, continuous integration and feedback, monitoring, operation, and continuous feedback. If you take a look at figure 3 you can see that what’s commonly known as agile is taking place on the left side of the lifeloop. The core value behind a

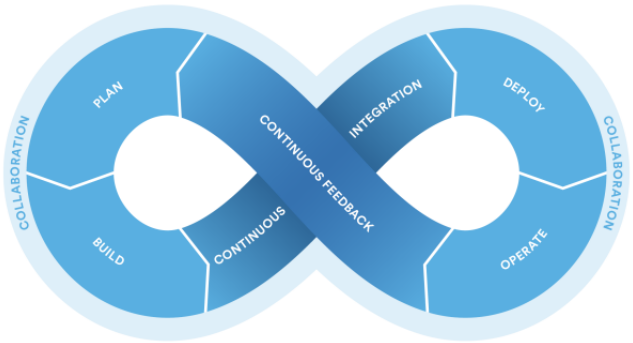

Fig. 3. The LifeLoop of DevOps (13)

well-functioning DevOps culture is collaboration throughout the entire lifeloop of a project. Collaboration is also a core value in BDD. So, let’s make BDD and DevOps collaborate! Yes, BDD is an agile practice, but using DevOps and Agile practices in tandem can turn out to be very successful. It helps in streamlining the development process within a team, department, or even an entire company which enables projects to be more adaptable to constant changes, at the same time being more effective.

VI. BDD as DevOps practice

One reason why BDD could improve the DevOps pipeline is that DevOps should not only focus on continuous integration and deployment (CI/CD) but also consider communication and culture to be able to make that shift (14). When BDD is implemented properly, it can prevent social friction(culture) and misunderstandings(communication), not only in a development team but in an organization as a whole.

Another reason is that when working in the continuous integration (CI) process, a high level of coordination and fast communication between the different units of the lifeloop is required. DevOps encourages everyone to contribute to the loop and do this where they are needed the most. Meaning that a developer can also configure deployments, operation engineers can also add test cases, and quality engineers can also configure the tests into the CI process (15). Having BDD-based tests running as a part of your CI process, the stakeholders will also be able to check the current state of the project (what is currently implemented and what is working and passing the acceptance tests) by themselves, since it is in a lingo that is easy to understand even for less technical people. Collectively, everyone in the lifeloop of DevOps is responsible for all aspects of the deliverable. For this wide collaboration to be possible, a shared language is necessary, hence following BDD is an easy way for successful DevOps projects.

VII. BDD Tools and Frameworks to use in DevOps

Many of the tools and frameworks that normally are used for BDD in the Agile world can be tweaked or directly adapted in

[1] When describing the life cycle for DevOps projects, this essay will use the word ”lifeloop” as a merge between life cycle and infinity loop

the DevOps culture. Worth mentioning is that just because you use a tool that is connected to BDD, does not automatically mean that you are practicing BDD. Below are examples of tools and frameworks from the different stages of BDD that can be adapted to work in DevOps.

A. Discovery Phase

As the discovery phase aims to overcome the blind spots of the "unknown-unknowns" in software development and to reduce the bottleneck of learning, the "Three Amigos" (described earlier) is often the preferred to-go-to tool (16). To make BDD work for DevOps, the Three Amigos should be tweaked to include more amigos, four (including Operation) or more.

B. Formulation Phase

Gherkin is the most common syntax used in BDD to formulate the scenarios discovered in the previous phase. If the discovery phase has been well structured, including people from the operation and development side of the process, Gherkin can be similarly applied in DevOps as it normally is used in BDD (17). A Gherkin syntax example with **Given**, **When**, and **Then**:

```
Scenario:   One discovered determinable
                        business situation
    Given   a precondition
      And   an other precondition
     When   the user make this action
      And   this other action
      And   yet this action
     Then   this testable outcome
                           is achieved
      And   something else we can test
                           happens too
```

(18)

C. Automation Phase

The Automation phase is where the scenarios formulated in the previous step get turned into executable code. Frameworks like Gauge or SpecFlow are typical BDD mechanisms for that(19; 20). This phase in the Agile world focuses on how to write automated tests from the executable code. To tweak this phase to work well in the DevOps culture would eg. be to provide the test results in a way that can be picked up by a CI tool such as Jenkins or GitLab (21; 22).

VIII. Conclusion

Besides ensuring that you deliver the right thing, BDD is a methodology that improves teamwork by reinforcing collaboration and communication all over a project's life cycle. Although it comes from the Agile world, it can easily be tweaked to also work well in a DevOps culture to improve the deliverable and team dynamics. BDD is not a silver bullet but used correctly and for the right projects, it can benefit both the quality of the product and the speed of development.

References

[1] A. Shannon, "What do you lose when teamwork fails?" June 2017. [Online]. Available: https://www.kent.edu/yourtrainingpartner/what-do-you-lose-when-teamwork-fails

[2] C. Drumond, "Is the agile manifesto still a thing?" [Online]. Available: https://www.atlassian.com/agile/manifesto

[3] K. Beck and et al ., "The agile manifesto," 2001. [Online]. Available: https://agilemanifesto.org/principles.html

[4] T. Hall, "Agile vs. devops," 2016. [Online]. Available: https://www.atlassian.com/devops/what-is-devops/agile-vs-devops

[5] A. Petäjävaara, "Dispelling inertia towards behavior-driven development: An assessment tool for development practice readiness," Ph.D. dissertation, KTH, School of Electrical Engineering and Computer Science (EECS)., Stockholm, Sweden, 2019.

[6] "Fig. 1. bdd vs tdd simplified life cycle." [Online]. Available: https://miro.medium.com/max/1456/1*3YramDu6HOAyz8OyDm9D0w.png

[7] "Gherkin syntax." [Online]. Available: https://cucumber.io/docs/gherkin/

[8] "Fig. 2. the different phases of bdd." [Online]. Available: https://d112uwirao0vo9.cloudfront.net/wp-content/uploads/2020/08/bdd-practices-flow.png

[9] Behaviour-driven development. [Online]. Available: https://cucumber.io/docs/bdd/

[10] "What are the three amigos in agile?" Mar 2021. [Online]. Available: https://www.agilealliance.org/glossary/three-amigos

[11] "Three amigos," May 2020. [Online]. Available: https://specflow.org/bdd/three-amigos/

[12] G. Dinwiddie, "If you don't automate acceptance tests?" 2009. [Online]. Available: http://blog.gdinwiddie.com/2009/06/17/if-you-dont-automate-acceptance-tests/

[13] "Fig. 3. the lifeloop of devops." [Online]. Available: https://3kllhk1ibq34qk6sp3bhtox1-wpengine.netdna-ssl.com/wp-content/uploads/devopsloop-600x325-1.png

[14] T. Goldberger, "Behavior-driven development in devops," May 2018. [Online]. Available: https://www.nagarro.com/en/blog/behavior-driven-development-in-devops

[15] A. Deshpande, "Devops testing tutorial: How devops will impact qa testing?" Mar 2021. [Online]. Available: https://www.softwaretestinghelp.com/devops-and-software-testing/

[16] "Discovery," May 2020. [Online]. Available: https://specflow.org/bdd/discovery/

[17] "Gherkin," Oct 2020. [Online]. Available: https://specflow.org/bdd/gherkin/

[18] "writing features - gherkin language." [Online]. Available: https://docs.behat.org/en/v2.5/guides/1.gherkin.html

[19] "Open source test automation framework." [Online]. Available: https://gauge.org/

[20] B. Dijkstra, "Bdd, specflow and the specflow ecosystem,"

Oct 2019. [Online]. Available: https://blog.testproject.io/2019/10/08/bdd-and-the-specflow-ecosystem/
[21] “Jenkins - build great things at any scale.” [Online]. Available: https://www.jenkins.io/
[22] “Gitlab is the open devops platform.” [Online]. Available: https://about.gitlab.com/