Success Factors and Challenges in Behavior Driven Development

Ivan Alexander, Nanda Adytiansyah, Oei Kurniawan Utomo, Sembada Denrineksa Bimorogo, Erick Pinenka

Abstract*: Behavior Driven Development (BDD) is a software development process that combines the general techniques and principles of Test Driven Development (TDD) with ideas from Domain Driven Design (DDD) and Object Oriented (OO) analysis. It describes a cycle of interactions with well-defined outputs, resulting in the deliverable, tested working software. Today, BDD has evolved into an established agile practice. However, compared to other agile methodology frameworks, such as Scrum and Kanban, BDD is a relatively new. Thus, available resources explaining BDD is still limited and the BDD approach is still under development. Based on this observation, this literature review aims to provide the key of success as well as the challenge that lies on the implementation process of BDD in IT Project. We identified 3 success factors and 5 challenges. The success factors are focusing in product value, having a thorough system behavior definition, and using the right BDD supporting tools. Meanwhile, the most challenging part are the difficulties in writing BDD scenario and automating the test case to maintain the system quality.

***Keywords*: Behavior Driven Development, BDD, IT Project, IT Software**

I. INTRODUCTION

Behavior Driven Development (BDD) was originally developed by Dan North to deal with the problems in Test Driven Development (TDD) and Acceptance Test Driven Development (ATDD) [1]. By using TDD and ATDD, the stakeholders' requirements are presented in form of test cases and all test cases are automated [2]. However, both TDD and ATDD approach doesn't tell the developer about why a user story is created and where to start writing a test, so developers tend to confuse about the test they should make. One common pitfall in implementing TDD and ATDD are the developers are focused on verifying the state of the product rather than the product's expected behavior and the test cases are hard to maintain because it is written in plain programing language [3]. The main goal of BDD is to get an executable behavior of a system. BDD approach incorporates aspects of requirements analysis, requirements documentation and communication, and automated acceptance testing. BDD is generally regarded as the evolution of TDD and ATDD; one of the improved aspects in BDD is the acceptance tests are clearly written and easily understandable. The behavior of a system is defined in a domain-specific language; a common language that reduces ambiguities and misunderstandings. This is further enhanced by including terms from the business domain in the domain-specific language that helps stakeholders to specify their tests. There are various framework supporting BDD, such as Cucumber and JBehave which can create abstraction of ubiquitous language to the test written in one programming language. The BDD approach in IT Project helps defining the desired business outcomes as behavior in ubiquitous language of DDD (Domain Driven Design) language. By doing so, the whole project teams are able to understand what are the business and technical challenge they might encounter. Basically, with BDD you focus on getting a clear understanding of the software behavior you are aiming for by communicating with stakeholders. Developers keep their focus on why a code should be created instead of putting too much emphasis on technical details. By implementing BDD, the team will have strong collaboration and high project visibility. The delivered software will meets all the predefined business behavior, hence will meet the user's expectation. The DDD ubiquitos language will help increasing the understanding and code readability for further software development and enhancement. The BDD scenario is written in Fig. 1. The scenario must follow the given-when-then template in BDD. Fig. 2 shows the translated scenario into integration/unit test in specific programming language. The method name has to follow the scenario, which increase code readability and maintainability in the development process. According to Carvalho et al., BDD is a specification technique that automatically verify all functional requirements by source code, using the connection of textual description of system behaviors as requirements and automated tests [4], [5]. Carvalho et al. stated that BDD allows reducing the risks and effort in implementing a system's specification by automated test of accepted system behavior in the 'then' clause, hence preventing the project to fall into the Boehm's cost of change curve, where the cost of change in a project increases through the time exponentially. Diepenbeck et al. stated the same thing about the highly coupled valuable link between the specification and the implementation of system behavior into automated test in BDD [6].

Revised Manuscript Received on June 22, 2020.

Ivan Alexander, Faculty of Engineering, Computer Engineering Department, Bina Nusantara University, Jl KH Syahdan 9, Jakarta 11480, Indonesia

Nanda Adytiansyah, Graduate Program, Master of Information Technology, Bina Nusantara University, Jl KH Syahdan 9, Jakarta 11480, Indonesia

Oei Kurniawan Utomo, Graduate Program, Master of Information Technology, Bina Nusantara University, Jl KH Syahdan 9, Jakarta 11480, Indonesia City

Sembada Denrineksa Bimorogo, Graduate Program, Master of Information Technology, Bina Nusantara University, Jl KH Syahdan 9, Jakarta 11480, Indonesia

Erick Pinenka, Graduate Program, Master of Information Technology, Bina Nusantara University, Jl KH Syahdan 9, Jakarta 11480, Indonesia

During the design and development process, the scenarios are ported step by step to executable tests. Compared to other agile methodology frameworks, such as Scrum (1993) [7] and Kanban (1940) [8], BDD is a relatively new (2006) [1]. Thus, available resources explaining BDD is still limited and the BDD approach is still under development [3]. Based on this observation, this literature review aims to provide the key of success and challenges in implementing BDD in IT Project.

```
Feature: user can retrieve his/her username
  Scenario: user makes call to GET /username
    Given the user is registered in the system
    When the user calls /username
    Then the user receives status code of 200
    And the user receives his/her username
```

Fig. 1.BDD scenario example

```
@When("^the user calls /username$")
public void the_user_calls_GET_username(){
    executeGet("http://localhost:8080/username");
}

@Then("^the user receives status code of 200$")
public void the_user_receives_status_code_of_200(){
    HttpStatus currentStatusCode = latestResponse
                                   .getTheResponse().getStatusCode();
    assertThat(currentStatusCode.value(), is(200));
}

@And("^the user receives his/her username$")
public void the_user_receives_hisher_username(String username){
    assertThat(latestResponse.getBody(), contains(username));
}
```

Fig. 2.BDD test case implementation example

This literature review is organized as follows. Section 2 states the research methodology. The results are presented in section 3. Section 4 ends this paper with a conclusion and discussion.

II. RESEARCH METHODOLOGY

This literature review focused on answering these questions about BDD approach in IT projects (a) what are the success factor in implementing BDD in an IT project and (b) what challenges have been reported in implementing BDD. The keyword derived from the research question is `("behavior driven development" OR "behavioral driven development") AND (challenges OR project OR problem OR success OR criteria)`. The keyword is executed on the ScienceDirect, Springer, and IEEE English journal literature databases. The research stage is conducted by two reviewer, which is sequentially delivered in Fig. 3 based on Vallon et al. [9]. The research stage began with searching all English literatures based on defined keyword between 2013 and 2019, abstract analysis, and full text analysis.

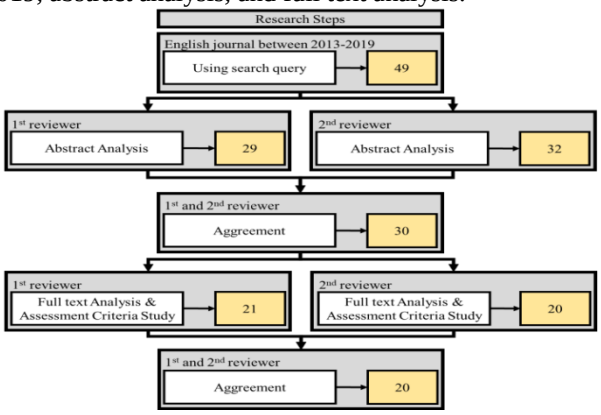

Fig. 3.Research steps

Error! Reference source not found.**Study Quality Assessment Criteria**

Criteria	Question
1	Does the study focus on BDD?
2	Does the study implement the BDD principle in the IT project?
3	Does the study mention the success factors in implementing BDD?
4	Does the study mention the challenges/problem in implementing BDD?

Error! Reference source not found.**Study Quality Assessment Criteria**

Literature Database	Number
Springer	12
ScienceDirect	7
IEEE	1

III. RESULT AND DISCCUSION

The answer for the research questions the success factors and challenges is presented in this section.

A. Success Factors

1) Focus on delivering valuable product

Melo et al. benefits from using the BDD point of view on the product behavior based on its value [10], [11]. Their customers became more satisfied with the frequent deliveries of valuable product which meet the market's need and the product owner will get fast feedback from the market using real-time survey results.

2) Define the product behavior thoroughly with examples

According to Zayan et al., examples are commonly used in software engineering testing process. Test cases should contain examples of what the software should do which is align with BDD scenario [12]. Palla et al. stated that a well-defined scenario will increase the development's pace because everyone in the team will have the same level of knowledge of the product they are developing. Therefore, asking questions focused on the behavior of the platform before and during the development stages, to avoid or at least reduce misunderstandings between stakeholders [13]. Alberola et al. also mentioned that an enterprise system needs methodologies that are appropriately defined to catch all the requirements for the design of systems, from the global system's purposes to the specification of the behavior of each individual entity [14].

3) Use supporting tools for development and testing

To derive executable test cases from DSL scenario, every step should be mapped into an executable test step [15]. Converting the scenario manually will need resource and time. The BDD project should use Cucumber or JBehave, to define acceptance tests [16], [17]. Development and testing tools helped experts in Australian project-AURIN to express workflows in a verbose, yet friendly language, and at the same time produce definitions that can be directly deployed into the toolbox composition and testing engine [18].

B. Challenges

1) The BDD's Domain Specific Language is difficult

Many Product Owners of some project are reported find difficulties to write the DSL scenario [10], [11].

Therefore, the development cycle could be delayed due to unclear specification by the stakeholders or Product Owner. Clear and concise scenario has the advantage of removing unnecessary technical gap and minimize amount of scenario-to-code mapping [19].

2) The DSL helps designing good behavior as acceptance criteria, but not high-quality code

The Product manager studied by Pyshkin et al. have problem in the code quality. Bad code quality leads to poor maintainability, although the acceptance test stories in BDD is written as narratives which bridge the product knowledge between the business and development team [20].

3) All DSL scenario as test case must be able to run automatically

Arcaini et al. stated that the BDD scenarios may be thought as use cases that drive the development and the development process should not be stopped until all the scenarios are executed without failures [21]. Lubke et al. and Willuweit et al. challenge the BDD team to make the product delivery seamless and can be done anytime which means all test scenarios should be run automatically [22], [23]. Meanwhile, having a team with such a high skill in automated testing is not easy [16] and manual tester should get an automation training [11].

4) The DSL scenario have to be documented carefully

The stakeholders must generate a set of behavior specifications that describes the whole system [24]. Meanwhile, different stakeholder will have varying product values [25]. Documented scenario will help the product owner and stakeholder to track the evolving value in a running project [16].

5) System integration testing should not be ignored

Watanabe et al. reported a failed project prototype using BDD [26]. Although all system's behavior is defined thoroughly, it did not guarantee that the webpage itself is covered against accessibility errors. The test cases only cover the functionality specified in them, which means unexpected behaviors of user also need to be defined. We have identified 3 success factors and 5 challenges in implementing BDD method. There are 9 papers that define the success factors. The success factors focus in defining the system behavior correctly and use the supporting tools as well as frameworks to help the development cycle with BDD. Meanwhile, there are 11 papers give us lists of challenges in implementing BDD in project. It means that it is not easy to adopt BDD, especially in automating the test and writing the BDD scenario. The automated test in BDD enforce the development team to have highly skilled software testing engineer. Also, the new BDD scenario idea for describing the system behavior with example makes it difficult for product owners and/or business analysts to define it clear and concise. Yet, once it is defined correctly, the development cycle's pace can be increased while maintaining the system quality with the automated testing.

IV. CONCLUSION AND FUTURE WORK

We presented a systematic literature review of implementing BDD in IT project. We analyzed 20 papers from 3 literature database since 2013, presenting success factors and challenges, which we summarized into 3 and 5 points respectively. The successful BDD implementation can be achieved if the product owner focus on the product value, the system quality is maintained in behavior scenario, and the supporting tools in the development cycle is correctly used. However, defining the system behavior correctly in form of BDD scenario is not easy and the development team should be able to automate all the test with the supporting testing tools. Also, the code quality, the scenario changes, and system integration testing should not be forgotten. Future works in this BDD research area could be done in exploring methods to overcome the challenges in implementing BDD. This can be useful for product owner and development team to design valuable and high-quality product.

REFERENCES

1. D. North, "Introducing BDD," Better Software, March, 2006.
2. W. Bissi, A. G. Serra Seca Neto, and M. C. F. P. Emer, "The effects of test driven development on internal quality, external quality and productivity: A systematic review," Information and Software Technology. 2016.
3. C. Solís and X. Wang, "A study of the characteristics of behaviour driven development," in Proceedings - 37th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2011, 2011.
4. R. A. De Carvalho, R. S. Manhães, and F. L. D. C. E. Silva, "Filling the Gap between Business Process Modeling and Behavior Driven Development," New York, 2010.
5. R. A. De Carvalho and R. S. Manhaes, "Mapping Business Process Modeling constructs to Behavior Driven Development Ubiquitous Language," arXiv10064892v1, 2010.
6. M. Diepenbeck and R. Drechsler, "Behavior Driven Development for Tests and Verification," in Formal Modeling and Verification of Cyber-Physical Systems, Wiesbaden: Springer Fachmedien Wiesbaden, 2015, pp. 275–277.
7. J. Sutherland, "Agile development: Lessons learned from the first scrum," Cut. Agil. Proj. Manag. Advis. Serv. …, 2003.
8. M. Olausson, J. Rossberg, J. Ehn, and M. Sköld, "Kanban," in Pro Team Foundation Service, Berkeley, CA: Apress, 2013, pp. 91–99.
9. R. Vallon, B. J. da Silva Estácio, R. Prikladnicki, and T. Grechenig, "Systematic literature review on agile practices in global software development," Inf. Softw. Technol., 2018.
10. C. de O. Melo et al., "The evolution of agile software development in Brazil," J. Brazilian Comput. Soc., vol. 19, no. 4, pp. 523–552, Nov. 2013.
11. V. Garousi and J. Zhi, "A survey of software testing practices in Canada," Journal of Systems and Software. 2013.
12. D. Zayan, A. Sarkar, M. Antkiewicz, R. S. P. Maciel, and K. Czarnecki, "Example-driven modeling: on effects of using examples on structural model comprehension, what makes them useful, and how to create them," Softw. Syst. Model., pp. 1–27, Jan. 2018.
13. P. Palla, G. Frau, L. Vargiu, and P. Rodriguez-Tomé, "QTREDS: A ruby on rails-based platform for omics laboratories," BMC Bioinformatics, 2014.
14. J. M. Alberola, V. Botti, and J. M. Such, "Advances in infrastructures and tools for multiagent systems," Inf. Syst. Front., vol. 16, no. 2, pp. 163–167, Apr. 2014.
15. G. Karagöz and H. Sözer, "Reproducing failures based on semiformal failure scenario descriptions," Softw. Qual. J., 2017.
16. E. Bjarnason, M. Unterkalmsteiner, M. Borg, and E. Engström, "A multi-case study of agile requirements engineering and the use of test cases as requirements," Inf. Softw. Technol., 2016.
17. S. Mäkinen et al., "Improving the delivery cycle: A multiple-case study of the toolchains in Finnish software intensive enterprises," Inf. Softw. Technol., vol. 80, pp. 175–194, Dec. 2016.
18. R. O. Sinnott and W. Voorsluys, "A scalable Cloud-based system for data-intensive spatial analysis," Int. J. Softw. Tools Technol. Transf., 2016.
19. T. Mens, A. Decan, and N. I. Spanoudakis, "A method for testing and validating executable statechart models," Softw. Syst. Model., 2018.

20. E. Pyshkin, “In the right order of brush strokes: a sketch of a software philosophy retrospective,” Springerplus, vol. 3, no. 1, p. 186, Dec. 2014.
21. P. Arcaini, A. Gargantini, and E. Riccobene, “Rigorous development process of a safety-critical system: from ASM models to Java code,” Int. J. Softw. Tools Technol. Transf., 2017.
22. S. Willuweit and L. Roewer, “The new y chromosome haplotype reference database,” Forensic Sci. Int. Genet., 2015.
23. D. Lubke and T. Van Lessen, “Modeling Test Cases in BPMN for Behavior-Driven Development,” IEEE Softw., 2016.
24. Á. Carrera, C. A. Iglesias, and M. Garijo, “Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development,” Inf. Syst. Front., 2014.
25. S. G. Yaman et al., “Introducing continuous experimentation in large software-intensive product and service organisations,” J. Syst. Softw., vol. 133, pp. 195–211, Nov. 2017.
26. W. M. Watanabe, R. P. M. Fortes, and A. L. Dias, “Acceptance tests for validating ARIA requirements in widgets,” Univers. Access Inf. Soc., 2017.

AUTHORS PROFILE

Ivan Alexander, works as Laboratory Officer at Binus University (Jakarta, Indonesia) in Computer Engineering Department. E-mail: ialexander@binus.edu

Nanda Adytiansyah, works as government employee who assigned in Constitutional Court of The Republic of Indonesia as Coordinator Infrastructur, Network, and Communication Department. He received his Master’s Degree at BINUS University, Jakarta, Indonesia. E-mail: nanda.adytiansyah@binus.ac.id

Oei Kurniawan Utomo, works as Software Engineer at DANA Indonesia. He received his Master’s Degree at BINUS University, Jakarta, Indonesia. E-mail: oei.utomo@binus.ac.id

Sembada Denrineksa Bimorogo, is currently pursuing a master’s degree program in computer science in Bina Nusantara University, Indonesia. E-mail: sembada.bimorogo@binus.ac.id

Erick Pinenka, is currently pursuing a master’s degree program in computer science in Bina Nusantara University, Indonesia. E-mail: erick.pinenka@binus.ac.id