A Literature Review of Behavior Driven Development using Grounded Theory

Abigail Egbreghts
University of Twente
P.O. Box 217, 7500AE Enschede
the Netherlands
a.b.egbreghts@student.utwente.nl

ABSTRACT

Behavior Driven Development (BDD) was first introduced as a solution to the issues that could be found in Test Driven Development (TDD). Today, BDD has evolved into an established agile practice. Despite it being an established practice, it is shown in this paper that the academic literature of BDD is still limited. This paper presents a review of the existing literature on BDD. The goal of this paper is to increase the understanding of BDD and to pinpoint gaps in the literature on BDD practices. As a result of this literature review, an overview of BDD concepts is given and explained, the use of BDD in software development according to literature is discussed, and finally two conceptual models are compared. This review can be used by researchers as a ground for further empirical research.

Keywords

Behavior Driven Development, Software development, literature review, Grounded Theory method.

1. INTRODUCTION

Many software development companies are transitioning from the "waterfall" model to "agile" practices. These agile software development methods include Scrum, Kanban, DevOps, TDD, and BDD. In the late twentieth century, the elaboration of the concept of "test first" into TDD occurred. TDD is a programming method that executes in short, repetitive cycles, wherein each cycle tests are written before the code [42]. Years later, Dan North invented BDD to transform TDD into a more efficient software development process [35]. The main goal of BDD is to get executable and well-defined specifications of software [35]. According to Dan North, the definition of BDD is: "*A second-generation, outside-in, pull-based, multiple-stakeholder, multiple-scale, high-automation, agile methodology*" [34]. In other words, BDD combines general techniques and principles of TDD with ideas from domain-driven design and object-oriented analysis and design. This combination is done in order to provide software development and management teams with shared tools and a shared process to collaborate on a software development. Table 1 presents a side-by-side comparison of the executions of TDD and BDD.

27th Twente Student Conference on IT, July 7, 2017, Enschede, The Netherlands.

Table 1: TDD and BDD execution comparison [42].

Phases	TDD	BDD
1	Write a test	Write a test scenario
2	Run the test and check if it fails	Execute the scenario and check if it fails
3	Write a code sufficient for the test to pass	Write a code sufficient to implement the expected behavior
4	Run the test and check if it passes	Execute the scenario and check if it passes
5	Refactor the code	Refactor the code

BDD is an emerging practice that has limited literature in comparison to the other agile methods. Moreover, there is a confusion about what BDD is and how it should be used. Therefore, it would benefit from the exposure of the potential theoretical foundations by reviewing all relevant literature on BDD. This literature review will be conducted according to the Grounded Theory. Through this literature review, theoretical foundations for BDD are proposed and will lead to a better understanding of BDD in general.

This paper continues in this Section with a description of the research problem. Section 2 includes a description of some related work, where literature of agile methods is reviewed. In Section 3, an explanation of the method of research is described. Furthermore, in Section 4 and 5, the taxonomy of BDD and the description of each concept can be found. Lastly, this paper presents in Section 6 and 7 the discussion and the conclusion of this research.

1.1 Problem Statement

As stated before, BDD is an emerging method and has become a widely used modern software development method [23]. However, it is the least researched out of the methods mentioned earlier. A quick literature search of the popular agile methods shows this in Table 2. For this search, the five popular software development methods were searched in the database of Scopus.

Table 2: Comparison of search results between methods.

Agile methods	#results
Scrum	2345
Kanban	1516
DevOps	248
TDD	644
BDD	65

The limited literature leads us to believe that there are gaps in the academic literature that have to be filled. To help fill in this gap, this research will analyze all relevant articles about BDD.

2. RELATED WORK

Various researchers have reviewed the literature on agile software development methods in general. For example, in 2002, Abrahamsson began filling the gap on agile methods by reviewing existing literature. His goal was to define and classify all agile software development approaches, to analyze ten agile methods against criteria, and to compare the similarities and differences of these methods [32]. Years later, another literature review was done by Dybå to analyze the benefits and limitations of all agile methods [17]. This review provided a map of findings, within a common research agenda, that can be compared for relevance in other settings and situations.

Furthermore, some researchers have focused on specific agile methods. In 2014, a literature review was done on DevOps using the Systematic Mapping Study. It studied 26 articles, derived DevOps characteristics from it, and discussed the benefits and the effects of implementing DevOps on software development performance [18]. Besides DevOps, another popular software development method, TDD, was also studied. Here a literature review was done to demonstrate evidence about TDD's effectiveness [44].

However, for BDD, only one literature review was found. In 2011, the literature research was done to identify the characteristics of BDD. Due to the limited number of literature published about BDD and its practices at that time, the research was mainly focused on BDD's tools [50]. This research will be an extension of that research. On the contrary to the previous review on BDD, this literature review will focus only on published literature up till now.

3. METHOD OF RESEARCH

To review the literature, we implemented the Grounded Theory method. This Grounded Theory method is used to help guide and systematize the reviewing process for a more optimal outcome that contributes to the theoretical progress [54]. This method is similar to the Systematic Mapping Study used in the literature review from Section 2. A clear difference between these two methods is; in the Systematic Mapping Study, the definition of research questions is one of the essential process step and in the Grounded Theory method it is not necessary. This research is focused on the general overview of BDD and not on one specific area or question, therefore, the Grounded Theory method was chosen. This method consists of five stages: defining, searching, selecting, analyzing, and presenting.

In the first stage, we established the search terms and the search databases. The established search terms were: "Behavior driven development" and "Behaviour driven development". The databases selected for the search were IEEE Xplore, ScienceDirect, Web of Science, ACM Digital Library, and Scopus. These databases were selected based on suggested relevant sources in the literature databases for this discipline [51].

Subsequently, the search for BDD literature began. In the selected databases, the previously mentioned search terms were implemented and the articles found were gathered and the duplicates were extracted. These articles were categorized under each search database. The gathered articles from the five databases, extracting the duplicates, are in total 69. In Table 3, the amount of articles categorized in each database can be seen.

Table 3: Amount of articles per database.

Database	Articles
IEEE	16
ScienceDirect	5
ACM	8
Web of Science	16
Scopus	24

After this stage, we selected the relevant articles for this research. The articles considered for this review were (1) in English, (2) published in 2006 and onwards, and (3) implemented BDD in one or more of the development phases (See figure 1 for the phases). Furthermore, a backward reference search was done, were citations from the sample literature were checked, to determine if prior articles should be considered and these articles was searched on Google Scholar. A total 44 articles were sampled for this research and categorized by a database (See Table 4).

In the fourth stage, we studied this sample and noted excerpts for each paper. Then, with the use of open coding these articles were analyzed. Open coding consists of conceptualizing and articulating the aspects that can be found in the excerpts noted beforehand in every paper [53]. In the following Section, the result of the open coding is presented.

4. TAXONOMY

Through open coding, major concepts of BDD were highlighted in the excerpts and in the titles of each article (See Appendix A for the resulted excerpts for each article). Overall 6 major concepts were highlighted continuously; ubiquitous language, requirements, acceptance tests, tools, collaboration, and automation. Table 5 gives an overview of which concepts were applied to which article categorized by databases.

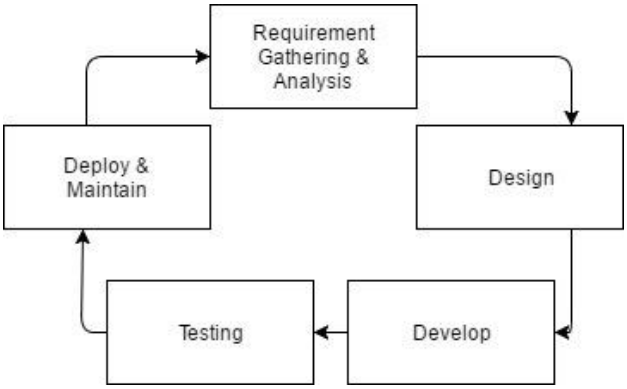

Figure 1: The system's development life cycle (SDLC) consist of Planning and Analysis, Design, Implementation, and Maintenance.

Table 4: Sample per database

Database	Sample amount
IEEE	12
ScienceDirect	4
ACM	6
Web of Science	12
Scopus	7
Google Scholar	3

Table 5: Concept Matrix

Database	Ref.	Ubiquitous Language	Requirements specification	Acceptance Tests	Tools	Collaboration	Automation
ACM	[2]	x	x	x		x	
	[3]		x	x			
	[6]		x	x			x
	[15]		x	x	x		
	[26]		x	x	x		x
	[33]		x	x	x		x
IEEE	[1]		x	x	x		x
	[9]	x	x		x		
	[20]			x	x	x	
	[22]			x	x		
	[38]		x	x		x	
	[39]	x	x	x			x
	[41]		x	x			
	[42]	x	x	x		x	
	[43]	x	x	x	x		x
	[45]		x	x			x
	[50]	x	x	x	x	x	x
	[52]		x			x	
Science Direct	[5]		x	x			
	[10]		x	x			
	[23]	x	x	x			
	[28]	x	x		x	x	
Web of Science	[5]	x			x		x
	[8]	x	x	x	x		x
	[21]	x	x	x			x
	[25]		x	x	x		x
	[27]		x			x	
	[29]		x	x	x		
	[31]	x	x	x		x	
	[36]		x				
	[37]		x				
	[40]			x			
	[47]		x	x			x
	[49]	x		x			x
Scopus	[4]		x	x	x		
	[11]	x	x	x			x
	[14]	x	x	x			
	[16]		x	x	x		
	[24]	x	x	x		x	
	[46]			x			x
	[48]		x	x			x
Scholar	[12]		x				
	[13]	x	x				
	[30]		x		x	x	

5. CONCEPTS

In this Section, each concept of BDD will be explained thoroughly. The data gathered by analyzing each article according to the concept will be used in this part for the explanation of each concept.

5.1 Ubiquitous Language

Ubiquitous language is discussed in sixteen papers out of the sample. An important conclusion drawn from all those papers is that the meaning of ubiquitous language is derived from the book written by Eric Evans named Domain-driven design (DDD). According to Evans, ubiquitous language is a familiar language, structured around the domain model, shared by both business and IT stakeholders to communicate the tasks connected to the software development [19]. The terms in the language are derived from the natural way of speaking, like whenever the clients try to explain to developers what for software they want, they use easy phrases, such as "when I do that, then this should happen" [43].

According to two papers, ubiquitous language is the fundamental core of BDD [50, 31]. Moreover, Häser et al. claim that it is essential for BDD's success since it can convey the domain concepts of the behavior as well as the expected results [23].

Almeida et al. state that the language serves two purposes, which are as project's documentation and as acceptance tests [2]. It helps in the analysis phase with the project's documentation, where all the stakeholders determine together terms derived from the requirements [13]. Following this phase, the terms from those requirements is captured into codebase [28], where they are designated to methods and classes in the acceptance tests that favors the definition of the system behavior [42].

Gil et al. express that implementing the creation of a language will have no impact on development process. Considering, everyone can get a clear idea of what is implemented and tested. This assures correctness of the defined testing strategy for a specific deployment [21].

Other benefits of ubiquitous language mentioned in the literature concerning the reduction of a gap between technical and business stakeholders [1, 11, 23, 31] are:

- Avoids faults and increase quality by forcing the stakeholders to reason systematically about the problem first [2].
- Enables stakeholders to actively participate in the definition of requirements [24].
- Less risk of miscommunication between stakeholders [26].

Considering all related papers, it is concluded that ubiquitous language makes codebase easier to understand, maintain and extend and provides faster and more efficient interaction. However, according to Soeken et al., there is always a risk of misunderstanding, due to ambiguity, by another designer or stakeholder that were not in the initial phase [49].

5.2 Requirements Specification

For most papers, the requirements specification concept is clearly described. This phase is what makes BDD distinctive from TDD. For those papers that were not marked in Requirements specification column in the concept matrix, we concluded that these papers focused on testing phase of BDD.

According to North, there are two predetermined specifications that are essential to BDD which are user stories and scenarios [35]. Firstly, a scenario is composed of several steps. A step is an abstraction that represents one of the elements in a scenario, which are contexts, events, and actions. The meaning of them is: in a particular case of a user story A, when event B happens, the answer to the system should be C (See template Scenario). One step is mapped to one test method [50]. Namely, each scenario composes of one test case, in which each sentence is referred to as a step [15]. Secondly, user stories are written to identify features and a feature is used to group a set of scenarios. BDD offers to users a template for both scenarios and user stories description.

Scenario	***User Story***
Given some initial context [A]. **When** an event occurs [B], **then** ensure some outcomes[C].	**As a** [X] **I want** [Y] **so that** [Z]

The goal of these requirements are to capture the needs of the customer, to map them into the code, and to ensure that the developers are developing the right software [4]. Which helps to guide the developers in knowing what to test as well as understanding what feature need to be implemented to pass tests and to write tests so that anyone can understand and read them [43].

5.3 Acceptance tests

Ten papers considers BDD a specification technique and do not acknowledge the testing phase of BDD. The definition of acceptance test in BDD, stated by the Dan North and in the papers, is that acceptance test is an executable specification which verifies the interactions and or behavior of the objects rather than the conditions and express the specification of behavior of the system [9, 50].

The process of acceptance tests starts with the development of the scenarios. Scenarios will be translated to tests which will drive the implementation. In order to pass a scenario, it is necessary that it passes all the steps. Each step follows the process of TDD which is "red, green, refactoring" to make it pass (See figure 2) [50]. The collection of scenarios forms an acceptance test suite for the system [33]. Acceptance tests are structured by means of features or feature files, where each feature can contain several scenarios [15]. As stated before this tests uses the terms from the specification to write methods, so that it is understandable for the stakeholders.

According to Bagheri and Ensan, the main goal of the developers within BDD is to write just enough code to make the tests pass; therefore, during the coding process, the developers rather focused on getting the desired functionality, than on and design and quality [3]. Hence, it is guaranteed that user stories

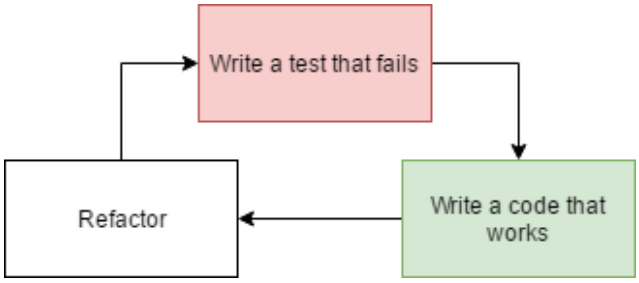

Figure 2: TDD "red, green, refactoring" derived from [50]

and scenarios, the needs of the customer, are satisfied and tested once a new feature is developed [3].

The test cases are usually encoded as scripts that can be interpreted and executed by an automated testing tool, this will facilitate regression testing [26]. According to King et al., challenges that testing can face are test comprehensibility, traceability, documentation, selection, and fragility [26].

5.4 Tools

Tool implementation and tool development are two of the categories the related papers are divided into among the sample. The tool implementation papers are about BDD tools that were implemented in a project and analyzed. From the tool implementation papers, it is concluded that BDD tools help with tying the requirements with the actual code of the system together and allows system requirement in plain text [43]. Yet, the tool development papers extend BDD for better collaboration and automation, each concepts are discussed in following Section 5.5 and 5.6.

In total 7 different tools were mentioned in the literature of BDD which are:

- Bumblebee: add-on to Junit that renders a readable document each time test suite is run [7].
- bUML: support all BDD activities that allows users to create models in UML and have a library for BDD [28].
- Cucumber: allows different views on tests, in which plain description of the test allows communication between stakeholders [15, 33, 42].
- Legend: a toolset that includes test automation tools and framework, programs and libraries for interacting with UI and persistent storage of system under test [26].
- PROD tool: generates testing codes as early as in the requirements engineering phase [20].
- Pyramid: support the employment of BDD in Python language [30].
- Rspec: support the refactoring of acceptance tests when restructuring production code [6, 42].

In spite of all these tools, Rahman and Gao stated that no tools exist for acceptance tests in BDD, which makes maintenance harder [39].

5.5 Collaboration

Out of the sample, thirteen paper discuss collaboration in BDD. A clear benefit of implementing BDD into a development project is to encourage collaboration among stakeholders, which includes the management, the developers, and the quality sectors [2, 27, 28, 30, 42]. They collaborate from the beginning of feature and story requirement analysis, with the creation of the specifications and identifying acceptance criteria [50]. What they do is specify in a ubiquitous language the expected behavior of a system for acceptance testing purposes [31] to increase communication and specification compression [8]. Collaboration makes it possible to generate models that are more consistent and proposes an exchange of knowledge using feedback obtained from the main stakeholders at the specification phase [52].

As stated in the previous Subsection, some tools and frameworks were created so that collaboration can be performed by stakeholders. Examples of these tools are Cucumber [42], Pyramid [30], and PROD tool [20]. Apart from the tools, cloud computing also helps with collaboration between stakeholders [20]. Cloud computing are servers on the Internet used for storage, management, and processing of data, which gives all stakeholder the opportunity to participate in the development. An example of a framework that was inspired by BDD is the Benchmark Driven Framework, an effective collaboration platform for researches [38].

5.6 Automation

Sixteen papers discuss the achievement of automation in BDD. Automation is achieved through the use of tools, frameworks, and test suites automation [35]. In BDD tools, all scenarios can be run automatically, which means when the acceptance criteria is imported and it is analyzed automatically. The classes implementing the scenarios will read the plain text scenario specifications and execute them. In other words, BDD allows having automatically executable plain text scenarios [50].
However, the automation process did not remain at scenarios as executable tests by a combination of structured natural language and system specific files [39]. Automated acceptance testing is the precursor to BDD's success. But according to Rahman, Gao, Borg, and Kropp, automated acceptance testing has its own challenges [6, 39]. Borg explained that the testing phase is the bottleneck of agile practices, where changes frequently happen because of the need to maintain test components to reflect on these changes [6]. In addition, Rahman and Gao explain that these tests may rely on expensive fixtures in databases, file systems, and network communications [39]. Challenges of this acceptance testing automation:

- Slow compared to unit tests [39]
- Suffer from non-determinism [39]
- Changing requirements[6]
- Restructuring production code[6]
- Restructuring acceptance tests[6]

In literature, these challenges of automation of acceptance tests are individually approached through the creation of tools [6]. In conclusion, according to the literature, automation achievement can occur through the use of tools for requirements specification and acceptance tests.

6. DISCUSSION

Previously little was found in the literature about a general overview of BDD. From this literature review, a concept matrix is developed. In this concept matrix, it is noticeable that some papers misses the two main concepts; acceptance tests and requirements specification. After the analysis of these papers, it is perceived that BDD can be defined as a test, specification, design or as a development technique. In Table 6, the amount and references of the papers are shown to which technique is applied in the papers.

Table 6: Literature based on techniques

Technique type	#of paper	Papers ref.
Test Technique	12	[8, 7, 10, 5, 22, 16, 40, 20, 29, 33, 26, 47, 42]
Specification Technique	15	[8, 23, 52, 28, 9, 20, 29, 11, 12, 13, 48, 27, 36, 47, 46]
Design Technique	5	[4, 30, 49, 37, 48]
Agile Development method	16	[43, 50, 6, 3, 31, 21, 2, 15, 14, 33, 39, 1, 41, 24, 25, 38, 45]

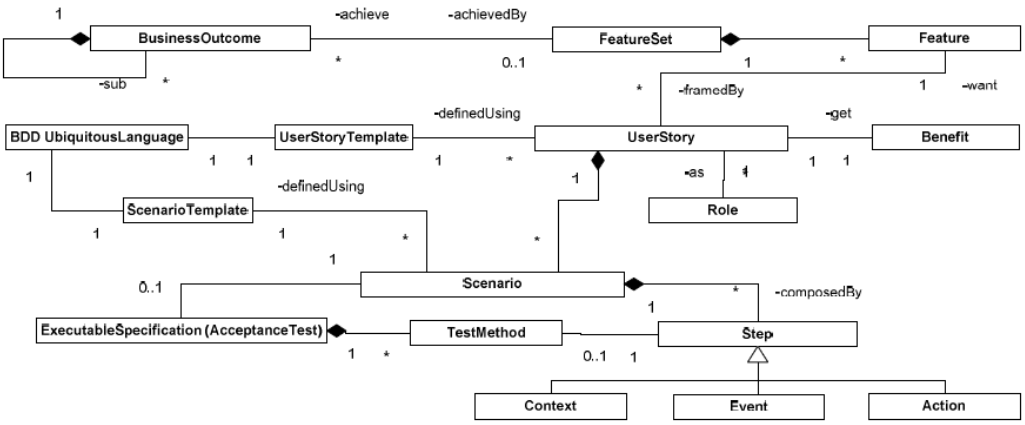

Figure 3: Conceptual Model from Solis and Wang [50].

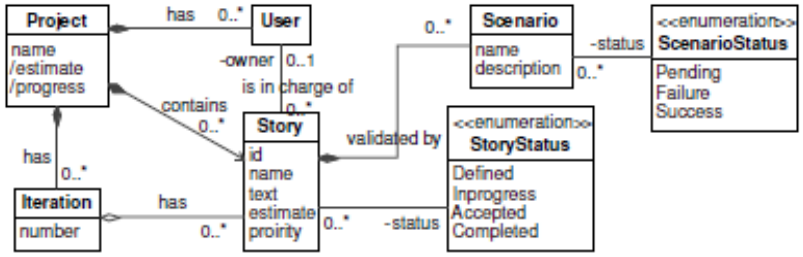

Figure 4: Conceptual model from Lazăr et al. [28].

Also from the literature review, two conceptual model for BDD are found (See figure 3 and 4). Through analyzing the conceptual model, it is clear that Solis and Wang identify BDD as a whole agile development method (Figure 3). On the contrary, Lazăr et al. regard it as a specification technique (Figure 4). Thus, this review proves that in literature, BDD is not a well-defined agile method. This inconsistency may be due to the fact that the concepts of BDD is not interlinked as a traditional agile development technique [50] and it is not embraced yet in projects. Furthermore, these findings may be somewhat limited by the literature found, there is a possibility that there are more concepts of BDD that are not in literature. However, these findings may help us understand the current state of BDD and how it is understood in literature.

7. CONCLUSION

According to Dan North, BDD is an agile method that combines several techniques and ideas from TDD and DDD to provide software development and team management through tools [35]. The aim of this present research was to examine the current literature with help of the Grounded Theory method to identify the concepts of BDD in literature. From this review, 6 main concepts were identified; ubiquitous language, requirements specification, acceptance tests, tools, collaboration, and automation. Furthermore, we noticed that in literature there are different definitions for BDD, among those are as a specification technique, design technique, testing technique, and as a whole, development method. These findings enhance our understanding of BDD concepts and its practices in literature. The present study confirms previous findings from Solis and Wang [50] about the concepts of BDD and contributes with additional evidence that suggests that in literature, BDD is not always implemented as a whole agile method. Nevertheless, in literature, all techniques do not have to be embraced to have development success.

The results of this review indicate future research into BDD as a whole agile development practice is strongly recommended. Future study could assess the long-term effects of implementation of BDD in practice. Additionally, this review shows that there is little knowledge of tools that support of the BDD acceptance test. Therefore, one future study could extend acceptance test tooling and the challenges that exist for acceptance test automation.

8. ACKNOWLEDGMENTS

Special thanks to my supervisor Lucas Meertens for providing me with his insights in this research and for his feedback throughout the whole research, and to my peer reviewer, Max Riesewijk, for his feedback on my draft paper.

9. REFERENCES

[1] Abrahamsson, P. et al. 2002. Agile software development: Review and Analysis. ESPOO. VTT Publications 478. 107p.

[2] Almeida, L., Cirilo, E., and Barbosa. E. A. 2016. SS-BDD: Automated Acceptance Testing for Spreadsheets. Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing (SAST). ACM, New York, NY, USA. 5. 10 pages. DOI= 10.1145/2993288.2993296

[3] Bagheri, E., and Ensan, F. 2013. Light-weight software product lines for small and medium-sized enterprises (SMEs). In Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research (CASCON '13). IBM Corp., Riverton, NJ, USA, 311-324.

[4] Bâillon, C., Bouchez-Mongardé, S., 2010. Executable requirements in a safety-critical context with Ada, Ada User Journal, 31 (2), pp. 131-135.

[5] Bjarnason, E., Unterkalmsteiner, M., Borg, M., and Engström, E. 2016. A multi-case study of agile requirements engineering and the use of test cases as requirements. Information and Software Technology. Volume 77, Pages 61-79, ISSN 0950-5849, DOI= 10.1016/j.infsof.2016.03.008.

[6] Borg, R., and Kropp, M. 2011. Automated acceptance test refactoring. In Proceedings of the 4th Workshop on Refactoring Tools (WRT '11). ACM, New York, NY, USA. 15-21. DOI= 10.1145/1984732.1984736

[7] Brolund, D. 2009. Documentation by example. Lecture Notes in Business Information Processing, 31 LNBIP. 251-252. DOI= 10.1007/978-3-642-01853-4_54

[8] Carrera, A., Iglesias, C.A., Garijo, M. 2014. Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development. Information Systems Frontiers. 16 (2). 169-182. DOI= 10.1007/s10796-013-9438-5

[9] Carter J. and Gardner, W. B. 2016. BHive: Towards Behaviour-Driven Development Supported by B-Method. IEEE 17th International Conference on Information Reuse and Integration (IRI), Pittsburgh, PA, 2016, pp. 249-256. DOI= 10.1109/IRI.2016.39

[10] Daylamani-Zad, D., Angelides, M. C., Agius, H. 2016. Lu-Lu: A framework for collaborative decision making game. Decision Support Systems, Volume 85, Pages 49-61, ISSN 0167-9236, DOI= 10.1016/j.dss.2016.02.011.

[11] De Carvalho, R.A., De Carvalho, E., Silva, F.L., Manhaes, R.S. 2012. Business Language Driven Development: Joining business process models to automated test. Advances in Enterprise Information Systems II - Proceedings of the 5th International Conference on Research and Practical Issues of Enterprise Information Systems, CONFENIS 2011, pp. 167-177.

[12] De Carvalho, R.A., De Carvalho, E., Silva, F.L., Manhaes, R.S. 2010. Filling the Gap between Business Process Modeling and Behavior Driven Development. Nucleo de Pesquisa em Sistemas de Informação (NSI), Instituto Federal Fluminense (IFF), Brazil, arXiv preprint arXiv:1005.4975.

[13] De Carvalho, R.A., De Carvalho, E., Silva, F.L., Manhaes, R.S. 2010. Mapping Business Process Modeling constructs to Behavior Driven Development Ubiquitous Language. arXiv preprint arXiv:1006.4892.

[14] Diepenbeck, M., Kühne, U., Soeken, M., and Drechsler, R. 2014. Behaviour driven development for tests and verification. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8570 LNCS. 61-77. DOI= 10.1007/978-3-319-09099-3_5

[15] Diepenbeck, M., Soeken, M., Grobe, D., and Drechsler, R. 2013. Towards automatic scenario generation from coverage information. In Proceedings of the 8th International Workshop on Automation of Software Test (AST '13). IEEE Press, Piscataway, NJ, USA, 82-88.

[16] Diepenbeck, M., Soeken, M., Grobe, D., and Drechsler, R. 2012. Behavior driven development for circuit design and verification. Proceedings - IEEE International High-Level Design Validation and Test Workshop, HLDVT, art. no. 6418237, pp. 9-16. DOI= 10.1109/HLDVT.2012.6418237

DOI= 10.1109/SEHC.2013.6602477

[17] Dybå T. and Dingsøyr T. 2008. Empirical studies of agile software development: A systematic review, Information and Software Technology. Volume 50, Issues 9–10, (August 2008). Pages 833-859, ISSN 0950-5849. DOI= 10.1016/j.infsof.2008.01.006.

[18] Erich, F., Amrit, C., and Daneva, M. 2014. Report: DevOps Literature Review. University of Twente, URL: https://www.utwente.nl/en/bms/iebis/staff/amrit/devopsreport.pdf [Accessed: 21-Mar-2017]

[19] Evans, E. 2004. Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional.

[20] Gao N. and Li, Z. 2016.Generating Testing Codes for Behavior-Driven Development from Problem Diagrams: A Tool-Based Approach. IEEE 24th International Requirements Engineering Conference (RE), Beijing, 399-400 DOI= 10.1109/RE.2016.54

[21] Gil, J.P., Garces, M., Broguiere, D., and Shen, T.-C. 2016. Behavior driven testing in ALMA telescope calibration software. Proceedings of SPIE - The International Society for Optical Engineering, 9913, art. no. 99130C, DOI= 10.1117/12.2232197

[22] Gohil, K., Alapati, N., and Joglekar, S. 2011.Towards behavior driven operations (BDOps). 3rd International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2011), Bangalore. 262-264. DOI= 10.1049/ic.2011.0095

[23] Häser, F., Felderer, M., Breu, R. 2016. Is business domain language support beneficial for creating test case specifications: A controlled experiment. Information and Software Technology. 79. 52-62. DOI= 10.1016/j.infsof.2016.07.001

[24] Hatko, R., Mersmann, S., and Puppe, F. 2014. Behaviour-driven development for computer-interpretable clinical guidelines. CEUR Workshop Proceedings, 1289. 105.

[25] Kamalakar, S., Edwards, S.H., and Dao, T.M 2013. Automatically generating tests from natural language descriptions of software behavior. ENASE 2013 - Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering. 238-245.

[26] King, T. M., Nunez, G., Santiago, D., Cando, A., and Mack, C. 2014. Legend: an agile DSL toolset for web acceptance testing. In Proceedings of the 2014 International Symposium on Software Testing and Analysis (ISSTA 2014). ACM, New York, NY, USA. 409-412. DOI= 10.1145/2610384.2628048

[27] Lai, S.-T. , Leu, F.-Y., and Chu, W.C.-C. 2014. Combining IID with BDD to enhance the critical quality of security functional requirements. Proceedings - 2014 9th International Conference on Broadband and Wireless Computing, Communication and Applications. BWCCA 2014. 7016084. 292-299. DOI= 0.1109/BWCCA.2014.78

[28] Lazăr, I., Motogna, S., and Pârv, B. 2010. Behaviour-Driven Development of Foundational UML Components. Electronic Notes in Theoretical Computer Science. Volume 264. Issue 1, (10 August 2010). 91-105. ISSN 1571-0661. DOI= 10.1016/j.entcs.2010.07.007.

[29] Li, N., Escalona, A., and Kamal, T. 2016. Skyfire: Model-Based Testing with Cucumber. Proceedings - 2016 IEEE International Conference on Software Testing. Verification and Validation. ICST 2016. 7515497. 393-400. DOI= 10.1109/ICST.2016.41

[30] Lopes Tavares, H., Guimarães Rezende, G., Mota dos Santos, V., Soares Manhães, R., Atem de Carvalho, R. 2010. A tool stack for implementing Behaviour-Driven

Development in Python Language. arXiv preprint arXiv:1007.1722

[31] Lopez-Pellicer, F.J., Latre, M.Á., Nogueras-Iso, J., Javier Zarazaga-Soria, F., Barrera, J. 2014. Behaviour-driven development applied to the conformance testing of INSPIRE web services. Lecture Notes in Geoinformation and Cartography. 325-339. DOI= 10.1007/978-3-319-03611-3_19

[32] Morrison, P., Holmgreen, C., Massey, A., and Williams, L. 2013.Proposing Regulatory-Driven Automated Test Suites. Agile Conference. Nashville, TN. 11-21. DOI= 10.1109/AGILE.2013.8

[33] Morrison, P., Holmgreen, C., Massey, A., and Williams, L. 2013. Proposing regulatory-driven automated test suites for electronic health record systems. 5th International Workshop on Software Engineering in Health Care (SEHC), San Francisco, CA. 46-49.

[34] North, D. 2009. How to sell BDD to the business. Skills Matter, London, URL: https://skillsmatter.com/skillscasts/923-how-to-sell-bdd-to-the-business#showModal?modal-signup-complete [Accessed: 17-Feb-2017]

[35] North, D. 2006. Introducing BDD, URL: http://dannorth.net/introducing-bdd [Accessed March 29, 2017].

[36] Okubo, T., Kakizaki, Y., Kobashi, T., Washizaki, H., Ogata, S., Kaiya, H., Yoshioka, N. 2014. Security and privacy behavior definition for behavior driven development. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 8892. 306-309.

[37] Oussena, S. and Essien, J. 2013. Validating enterprise architecture using ontology-based approach: A case study of student internship programme. 3rd International Symposium ISKO-Maghreb, art. no. 6728200, DOI= 10.1109/ISKO-Maghreb.2013.6728200

[38] Polikovsky, S., Quiros-Ramirez, M. A., Kameda, Y., Ohta Y., and Burgoon, J. 2012. Benchmark Driven Framework for Development of Emotion Sensing Support System. European Intelligence and Security Informatics Conference, Odense. 353-355. DOI= 10.1109/EISIC.2012.27

[39] Rahman, M., and Gao, J. 2015. A Reusable Automated Acceptance Testing Architecture for Microservices in Behavior-Driven Development. IEEE Symposium on Service-Oriented System Engineering. San Francisco Bay, CA. 321-325. DOI= 10.1109/SOSE.2015.55

[40] Rahman, M., Chen, Z., and Gao, J. 2015. A service framework for parallel test execution on a developer's local development workstation. Proceedings - 9th IEEE International Symposium on Service-Oriented System Engineering. IEEE SOSE 2015, 30. 7133524. 153-160. DOI= 10.1109/SOSE.2015.45

[41] Sackey D. J., and Ullmann, N. 2012. Visualizing data, encouraging change: Technical interventions in food purchasing. IEEE International Professional Communication Conference. Orlando, FL. 1-5. DOI= 10.1109/IPCC.2012.6408654

[42] Santos, E.C.S., Beder, D.M., and Penteado, R.A.D. 2015. A study of test techniques for integration with Domain Driven Design. In Information Technology-New Generations (ITNG). 12th International Conference. 373-378. IEEE. DOI= 10.1109/ITNG.2015.66

[43] Sathawornwichit, C., and Hosono, S. 2012.Consistency Reflection for Automatic Update of Testing Environment. IEEE Asia-Pacific Services Computing Conference. Guilin. 335-340. DOI= 10.1109/APSCC.2012.49

[44] Shull, F., Melnik, G., Turhan, B., Layman, L., Diep, M., and Erdogmus, H. 2010. What Do We Know about Test-Driven Development?. IEEE Software. 27. 6. 16-19, (Nov.-Dec. 2010) DOI= 10.1109/MS.2010.152

[45] Silva, T. R. 2016. Definition of a Behavior-Driven Model for Requirements Specification and Testing of Interactive Systems. IEEE 24th International Requirements Engineering Conference (RE). Beijing. 444-449. DOI= 10.1109/RE.2016.12

[46] Silva, T.R., and Winckler, M.A.A. 2016. Towards automated requirements checking throughout development processes of interactive systems. CEUR Workshop Proceedings, 1564.

[47] Silva, T.R., Hak, J.-L., Winckler, M. 2016. Testing prototypes and final user interfaces through an ontological perspective for behavior-driven development. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 9856 LNCS. 86-107. DOI= 10.1007/978-3-319-44902-9_7

[48] Sivanandan, S. and Yogeesha C.B. 2014. Agile development cycle: Approach to design an effective Model Based Testing with Behaviour driven automation framework. 20th Annual International Conference on Advanced Computing and Communications. ADCOM 2014 - Proceedings. 7103243. 22-25. DOI= 10.1109/ADCOM.2014.7103243

[49] Soeken, M., Wille, R., and Drechsler, R. 2012. Assisted behavior driven development using natural language processing. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 7304 LNCS. 269-287. DOI= 10.1007/978-3-642-30561-0_19

[50] Solis, C., and Wang, X., 2011. A Study of the Characteristics of Behaviour Driven Development. 37th EUROMICRO Conference on Software Engineering and Advanced Applications. Oulu. 383-387. DOI= 10.1109/SEAA.2011.76

[51] Universiteit Twente. 2017. Services and manuals. Available at: https://www.utwente.nl/en/lisa/service-manuals/!/product/p929099/subject-guide-business-it [Accessed 26 Jun. 2017].

[52] Wanderley, F. and da Silveria, D. S. 2012. A Framework to Diminish the Gap between the Business Specialist and the Software Designer. Eighth International Conference on the Quality of Information and Communications Technology. Lisbon. 199-204. DOI= 10.1109/QUATIC.2012.9

[53] Webster, J., and Watson, R.T. 2002. Analyzing the past to prepare for the future: Writing a literature review. MIS quarterly. xiii-xxiii.

[54] Wolfswinkel, J.F., Furtmueller, E., and Wilderom, C.P. 2013. Using grounded theory as a method for rigorously reviewing literature, European journal of information systems. 22(1).45-55.

APPENDIX

A. Excerpts

ACM Digital Library

REF	DESCRIPTION
[2]	Describes a BDD framework tool, Legend, aimed to streamlining functional testing in large-scale agile project.
[3]	Adopt and extend BDD methodology to equip developers with the mean to capture and manage software variability
[15]	An empirical study on coverage in BDD projects. The results are for two different implementations.
[26]	A testing approach, which supports the application of BDD concepts and ideas on the context of Spreadsheets is described and tested.
[33]	The creation and use of BDD acceptance test suites to enable organizations to compare their systems to regulation in a repeatable and traceable way.
[41]	Discuss the problems and challenges that rise through acceptance test written advance of to the production code(BDD) and developed a supporting refactoring plug in.

IEEE Xplore

REF	DESCRIPTION
[1]	BDD scenarios was used as the basis of an automated compliance test suite for standards.
[9]	Maps BDD scenarios with B-method to explore system behavior and exposing gaps in requirements.
[20]	Presents a tool inspired by BDD community provides a place where multiple stakeholders can draw.
[22]	An exploratory study on BDM was presented, use Cucumber for this study. Presents in the end the benefits of BDM.
[38]	Presents a new framework for development of emotion sensing support, inspired by BDD.
[39]	Address the concerns of implementing BDD frameworks into micro services and presents a solution which is reusable automated acceptance testing architecture.
[41]	Presents a mobile app where they use BDD to describe the needs of the consumers.
[42]	An exploratory study on integration of TDD and BDD with DDD to increase quality of a software.
[43]	Presents an approach for maintaining consistency, acceptance testing process, based on BDD.
[45]	Propose an approach based on BDD to support automated assessment and development.
[50]	A study where they identified six BDD characteristics, through toolkits analysis and limited literature, and showed how they are interlinked.
[52]	An interactive approach to the agile requirements modeling was presented, to contribute to communication between business and it, with use of BDD scenarios.

ScienceDirect

REF	DESCRIPTION
[5]	A case study to investigate how test case can support the main requirements activities.
[10]	A game LuLu was proposed with the key set ingredients and the performance assessment was done using BDD.
[23]	An empirical study on whether supporting the ubiquitous language characteristics as specified in BDD in beneficial compared to using the base language.
[28]	A UML profile was introduced to obtain a BDD framework for executable UML stories and scenarios.

Web of Science

REF	DESCRIPTION
[7]	Present an open source tool that addresses the shortcomings of documentation by extending BDD.
[8]	Presents a testing tool which automatically generates test cases skeletons from BDD scenarios specifications.
[21]	Adapted BDD to testing activities applied to TELCAL software.
[25]	Developed a tool called Kirby that uses natural language processing techniques, code information extraction and probabilistic matching to automatically generate executable software tests from structured English scenario descriptions.
[27]	To enhance security this paper combined IDD and BDD requirements.
[29]	Presents a model based testing tool that can automatically generate effective Cucumber test scenarios.
[31]	Explores the suitability of BDD approach to conformance testing.
[36]	A method for defining the acceptance criteria (BehaveSafe) for the BDD by creating a threat and countermeasure graph is proposed.
[37]	Experimental research-based study for modeling EA and the validation was done adopting the BDD method.
[40]	Presents a service framework for parallel tests execution, explains the needs and challenges for development
[47]	Proposes an approach based on BDD to support the automated assessment of artifacts along the development process of interactive systems.
[49]	Propose an assisted flow for BDD where the user enters into a dialog with the computer which suggests code pieces extracted from the sentences.

Scopus

REF	DESCRIPTION
[4]	Introduces a tool that is designed to bring BDD to Ada language.
[11]	Presents a method that aims to extend BDD, by connecting business process models directly to automated tests.
[14]	Presents a new approach based on BDD that combines testing and verification
[16]	A new design flow based on BDD is proposed and present a technique that allows automatic generalization of test cases.
[24]	An approach for specification and analysis of Clinical Guidelines that was inspired by BDD.
[46]	An approach for automating the requirements assessment using a BDD perspective.
[48]	A design of a behavior driven test automation framework using MBT and how it can be effectively used during Agile development is explored.

Scholar

REF	DESCRIPTION
[12]	Mapping BPM to BDD, proposes the use of Finite State Machines.
[13]	Provides a mapping from the basic constructs that form the most common BPM languages to Behavior Driven Development constructs. Extends [8]
[30]	Presents a tool for implementation, specification and test of software through BDD in Python language.