On the Empirical Evaluation of BDD Scenarios Quality: Preliminary Findings of an Empirical Study

Gabriel Oliveira, Sabrina Marczak
Computer Science School, PUCRS
Porto Alegre, Brazil
gabriel.pimentel@acad.pucrs.br,
sabrina.marczak@pucrs.br

***Abstract*—Behavior-Driven Development (BDD) is a set of software engineering practices which uses a ubiquitous language, one that business and technical people can understand, to describe and model a system by a series of textual scenarios. However, we believe that the value of those textual scenarios is connected with how well they convey and document the details discussed by the team about the behaviors needed to fulfill customer needs. Thus, there is the need to define what would be a "good" BDD scenario and how to evaluate scenarios. On this research design paper, we describe a study designed to understand how known quality attributes from other types of requirements specifications (like use cases or user stories) can be used with BDD scenarios. We also report our preliminary findings after performing it with novice evaluators. Those findings indicate that some of those attributes can be used to support BDD scenarios assessments.**

***Keywords*—documentation quality, documentation evaluation, behavior-driven development, empirical study**

I. Introduction

Behavior-Driven Development (BDD) is a set of practices that bring business analysts, developers, and testers together to collaboratively understand and define executable solution requirements in the form of scenarios. Smart [1] states that those scenarios common language allows an easy, less ambiguous path from end-user business requirements to the specification on how the software should behave, serving as a guide the developers in building a working software with only the features that are known to really matter to the business.

However, there is a lack of studies who evaluates how those scenarios are written. It is well known that bad requirements are one of potential causes of a project failure [2] and that bad scenarios documentation can lead to misleading information that will negatively impact the tests ability to reflect the system coverage and the team confidence on them [3].

We judge it necessary to better understand how we can prevent BDD scenarios, that brings many benefits to the development team, to suffer from those problems caused by bad documentation. To address that, a master's degree research is being performed, with the goal to propose guidelines on how to write "good" BDD scenarios. As a first step to support that goal, we organized an empirical study with post-graduate students to understand how novice evaluators use known quality criteria to judge the quality of BDD scenarios. Next, based on the list of identified criteria, we plan to validate them with industry practitioners (second step) and ask for recommendations on how to write "good" BDD scenarios (final step). This paper aims to highlight our efforts on the first step - the validation of a list of known quality criteria that would give us some insights to guide later discussions with practitioners about their criteria.

Therefore, this paper proceeds as follows: Section 2 reviews the concepts of Requirements, Use cases and BDD and reflects upon the different set of criteria to validate requirements. Section 3 presents the study design we performed to acquire a deeper understanding of how quality attributes could be used to validate BDD scenarios and the preliminary findings during of this study. Section 4 and 5 concludes this paper by summarizing our perceptions about the study and outlining some directions for our on-going research, respectively.

II. Background

A. Traditional Requirements

In this paper, we focus on solution requirements, which describes the capabilities of a solution and provide the appropriate level of details to allow the proper implementation of that solution [4]. More precisely, we focus on functional requirements, that describes the capabilities a solution must have in terms of system behaviors. A traditional format to describe system behavior is trough use cases.

Cockburn [5] says that a use case captures a contract between the stakeholders of a system about its behavior and describes the system's behavior under various conditions by interacting with one of the stakeholders (the *primary actor*, who wants to perform an action and achieve a certain goal). They are used to express solution requirements for software systems and can be put into service to stimulate discussion within a team about an upcoming system. Besides the primary actor, that interacts with the system, a use case consists of: the *scope* identifies the system that we are discussing, the *preconditions* and *guarantees* say what must be true before and after the use case runs, the *main success scenario* represents a case in which nothing goes wrong and the *extensions section* describes what can happen differently during that scenario.

B. Behavior-Driven Development

Most agile methodologies tend to not use traditional requirements or use cases, but represents requirements using user stories. For Cohn [6], a user story describes functionality that will be valuable to either a user or purchaser of a system or software. Lucassen et. al [7] summarize that user stories only

capture the essential elements of a requirement: *who* it is for, *what* it expects from the system, and *why* it is important.

Cohn [6] and Jeffries [8] state that a user story is composed of three elements. The Card, an expression of the essential elements of a requirement, represents customer requirements rather than document them - it has just enough information to identify the requirement and remind everyone what the story is about. The Conversation is an exchange of thoughts, opinions, and feelings - largely verbal but can be supplemented with documents. The best supplements are examples, representations of the Confirmation with the goal to let customers tell developers how she will confirm that they have done what is needed. That Confirmation, provided by those examples, is what makes possible the simple approach of Card and Conversation. When the conversation about a card gets down to the details, the customer and programmer specify what needs to be done in the format of acceptance tests.

Behavior-Driven Development (BDD) is an umbrella term that encapsulates a set of practices that uses scenarios as a ubiquitous language to describe and model a system as executable specifications [1]. Scenarios are expressed in a format known as Gherkin, that is designed to be both easily understandable for business stakeholders and easy to automate using dedicated tools. Each scenario is made up of a number of steps, where each step starts with one of a small number of keywords. The natural order of a scenario is *Given* a precondition *When* an action is performed *Then...* a result is observed, similar to the guarantees seen on use cases. For different actions, different scenarios should be created, an approach similar in purpose with the use case's extension.

It is a known fact that BDD scenario is a format to represent acceptance tests [9]. It fulfills the role of the Confirmation term defined by Jeffries [8] by specifying scenarios who convey the product behavior - thus, describing solution requirements.

C. Requirements Validation

Requirements validation is a phase on traditional requirements engineering process that is known to support the three other activities - requirements elicitation, requirements analysis and requirements specification - by identifying and correcting errors in the requirements [10].

The Business Analyst Body of Knowledge (BABOK) newest edition [11] states that while quality is ultimately determined by the needs of the stakeholders who will use the requirements or the designs, acceptable quality requirements exhibit many characteristics. It lists some characteristics a requirement must have in order to be a quality one, as follows: atomic, complete, consistent, concise, feasible, unambiguous, testable, prioritized, and understandable. A slightly different list is found on a prior version [4], as follows: cohesion, completeness, consistency, correction, viability, adaptability, unambiguity, and testability. Although the characteristics' meaning is defined, no measurement guidance is given. Cockburn [5] takes inspiration on those attributes to define rules on how to validate use cases. Those rules are summarized in a pass/fails questionnaire to be applied on use cases - good use cases are those that yield an "yes" answer to all of them.

For user stories, a format to represent agile requirements, the INVEST (Independent-Negotiable-Valuable-Estimable-Scalable-Testable) acronym presented by Cohn [6] seems to be one of the mostly used criteria in use by industry practitioners, as Heck and Zaidman [12] have stated on their practitioners interviews. Despite the INVEST acronym popularity, Heck and Zaidman [12][13] expand the use of criteria with a framework of their own, as they believe that the notion of quality for agile requirements is different from the notion of quality for traditional up-front requirements. Lucassen et. al [7] define additional criteria on top of Heck and Zaidman [13] to evaluate user stories on their QUS Framework. Those extensions to the INVEST acronym were tailored to be used by user stories only, a requirement format that lacks the detailed description found on BDD scenarios – therefore, we decided to use on our empirical study only the generic attributes found on the INVEST acronym and not those extensions.

Little attention is being given to the quality of the written documentation expressed on BDD scenarios format. BDD scenarios can be only evaluated based on characteristics taken from the Smart [1] experience, such as: scenarios steps expressiveness, focused on what goal the user want to accomplish and not on implementation details or on screen interactions (writing it in a declarative way and not on a imperative way); the use of preconditions on the past tense, to make it transparent that those are actions that have already occurred in order to begin that test; the reuse of information to avoid unnecessary repetition of words; and the scenarios independence.

Even though Smart [1] specifies a few examples of good and bad scenarios in order to demonstrate those characteristics he described, we feel it would be important to have a refined quality questionnaire built by the collective knowledge of other practitioners, similar to the one used on Cockburn's use cases [5]. Using Jeffreys [8] terms, we believe that, if the Confirmation in the form of BDD scenarios is not a good representation of the details discussed in the Conversations by the team and the customer, the simple and lightweight approach of writing customer needs on Cards is not effective.

III. Evaluating BDD Scenarios Quality

As we have no knowledge about a quality questionnaire to evaluate BDD scenarios, we must understand what criteria are important to evaluate the quality of BDD scenarios and from there build our own questionnaire. Therefore, our purpose is to find a list of quality attributes that would work with BDD scenarios, in a similar way that traditional attributes [4][11] work with requirements documents and INVEST[6] works with user-stories. We find inspiration on what similar studies did with Use Cases [5][14] and User Stories [12][7].

In preparation to creating that quality attributes list, we first seek to gain insight on whether the existing quality attributes would work with BDD scenarios. To that end, we organized a study with 15 post-graduate students to understand how they evaluate BDD scenarios. All of those students have previous experience working in the industry as developers, technical leads or even managers. They also have declared to know well how to write use cases, but not much about how to write user stories or BDD scenarios. The quality attributes used on the study were: atomic, complete, consistent, concise, estimable, feasible, independent, negotiable, prioritized, small, testable, understandable, unambiguous, and valuable.

However, we had three concerns before starting: (a) how a person's evaluation of a textual document using quality attributes would differ based on the representation format and (b) how would an evaluator experience with the product affect his analysis of a requirement and (c) how the use of traditional requirements attributes would be different from INVEST ones.

To address our first concern, we judged it necessary to compare how the same quality attributes would be used to evaluate BDD scenarios and Use Cases, in order to clarify a person's motives and analysis rationale. The choice of formats to compare was based on the fact that both, scenarios and use cases, seems to prefer a more declarative way to describe a behavior, rather than imperative, to stay detached from the implementation details [5][1]. Also a use case execution path seems to be a BDD scenario on a different format - on Cockburn's book [5], the main scenario is always described. Additionally, writing requirements represented as use cases was among the skills that all the students had.

To address our second concern, we judged it necessary to first ask the students to write requirements to two fictional products (PA and PB) in different domains, using the two requirements formats we have chosen, so they could feel more familiar with the product domain and the requirements formats before evaluating other students' work.

To address our third concern, we mixed traditional attributes from both BABOK editions [4][11] and the INVEST framework [6] in a single alphabetically ordered list. We let the scope of the analysis - if one should evaluate each scenario or use case separately or together with the others from the same feature - open to interpretation to measure this aspect.

A. Fictional Products

The products were presented to the students in a product vision board format [15]. On that occasion, they had the chance to ask questions, straighten their understanding on how to solve the business problem presented, and collaboratively draft a list of high-level requirements expressed as a use case scope format [5] and a user story format [5]. After this initial discussion, we took their suggestions and created the final high-level requirements. Finally, a new discussion round was performed to allow the students to read the final high-level requirements, validate them, negotiate their details and decide whether each one should be part of the scope or not. Those discussion sessions were planned to guarantee a common ground of understanding before producing use cases (based on the high-level requirements represented by scope statements) and BDD scenarios (based on high-level requirements represented by user stories). After the last round of discussions, the students then proceeded to perform their tasks in individual manners.

Product A aim was to develop a mobile app to help people with food allergy find places to eat free of ingredients that cause them allergy and distress. The user should be able to have an easy way to indicate which kind of food allergy or restrictions one has. There were two target groups: Users with any kind of food allergy that are quickly (e.g., while driving, walking, chatting with another person, etc) looking for a place to eat; and Customers, owners of restaurants, whose company business reputation would be improved if they used this new client search channel. Product B was a social-network website that aims to bring low cost book readers and resellers (who sell second-hand books) together. Readers would fill their profiles with genre interests and literature thoughts in exchange of badges, a higher fame status, and occasional promotions directed to them. Those social network interactions would provide data to resellers so they could direct their marketing efforts and promotions to the right subset of users.

TABLE I. SAMPLE OF STUDY ORGANIZATION FOR 4 STUDENTS (S1 TO S4)

ID	Task 1	Task 2	Task 3	Task 4
S1	Write UC for PA	Write US+BDD for PB	Evaluate US+BDD for PA *(S2 task 1)*	Evaluate UC for PB *(S4 task 2)*
S2	Write US+BDD for PA	Write UC for PB	Evaluate UC for PA *(S1 task 1)*	Evaluate US+BDD for PB *(S3 task 2)*
S3	Write UC for PA	Write US+BDD for PB	Evaluate US+BDD for PA *(S4 task 1)*	Evaluate US+BDD for PB *(S2 task 2)*
S4	Write US+BDD for PA	Write UC for PB	Evaluate UC for PA *(S3 task 1)*	Evaluate US+BDD for PB *(S1 task 2)*

B. Design Description

Therefore, we asked the students to perform the following sequential tasks: write requirements for PA with requirement format assigned; write requirements for PB with another requirement format assigned; evaluate other student requirements for PA and for PB. As explained before, each student was assigned a requirements format among user stories and BDD scenarios (US + BDD) or use cases (UC) to perform the writing tasks (Tasks 1 and 2). Therefore, we had virtually two group of students - Group 1 (G1) contained those who develop US+BDD for product A and UC for product B and Group 2 (G2) those who develop UC for product A and US+BDD for product B. For the evaluation tasks, students on one group should receive functional requirements from the other group. For example, students from G1 who developed BDD scenarios for product A should now evaluate the use cases for product A as part of Task 3. Task 4 should be executed in the same way, but considering product B. Table I exemplifies how four students would perform the tasks provided.

As can be seen on Table I, we made sure that the same student would not read two requirements formats from the same colleague when performing Tasks 3 and 4 in order to avoid a person's writing style to affect the evaluation of a scenario or use case. Also, it was desirable that the student gets the opposite requirement format to evaluate a given product.

IV. PERCEPTIONS ABOUT THE STUDY

The study has finished its execution phase, with all the 15 students having performed their four assigned tasks, but the analysis of the data collected throughout the completion of tasks 1 to 4 and the students interpretation about quality attributes will still be crossed with practitioners opinions about what makes a "good" BDD scenario. We could already take some degree of insight from our notes during the study and

from a final 2 hours long discussion round with the students once they concluded performing their work.

When inquired about comparing their work with different requirement formats, the students reported that BDD scenarios are easier to write (due to the lack of description of every user interactions with the system) but hard to analyze when compared to use cases. Due to that opinion, BDD scenarios description were found to be more negotiable than use cases (one of the INVEST attributes). In the other hand, use cases were found to be more easily testable. It's interesting to note their assumption that use cases descriptions need to be more imperative (with details explicitly defined) and scenarios descriptions need to be declarative (without implementation details) - one we did not foreseen.

Also, the list of chosen attributes generated confusion on the evaluation of BDD scenarios. As it mixed traditional requirements characteristics from both versions of the BABOK [4] [11] and the INVEST acronym provided by Cohn [6], some students did not understand how to properly differentiate some attributes when evaluating scenarios. For example, atomicity from BABOK [11] and independence from INVEST [6] were often seen as opposites, even if this is not always the case - a requirement that specifies only one action (atomic) can have many or none dependencies with others. One student reported that this confusion may have come from the difficulty to see bad examples of BDD scenarios, specially built to demonstrate when an attribute fail. They did not reported about problems to use INVEST acronym on use cases, though.

Some attributes may not make sense to evaluate a single BDD scenario - completeness may only represent how a set of scenarios cover a user story, for example. As it was said before, we let the scope of the analysis open to interpretation on purpose. Some students have joined the scenarios from a user story together before analyzing them, which raises questions on what was the rationale behind their decision.

The students also reported that inputs/outputs are important on BDD scenarios to help on prioritization and testability. As they don't have use cases' imperative description of every user interaction with the system to help them understand the technical steps needed to perform an action, stating inputs and outputs clearly may impact the ability of the team to estimate effort and analyze impact, thus impacting a requirement's testability. A few of them understands that the lack of inputs/outputs can be seen as a lack of the completeness attribute. Those perceptions resembles the ones on Smart's [1] book.

Lastly, written rigor was judged as necessary by the students to reinforce the team's common understanding during conversations with customers. As they stated, the writing of BDD scenarios is easier that use cases (due to the use of plain English language to represent behaviors and the declarative description format) and could be done by anyone - however, they reported that just someone who provides good examples would be desirable, in order to express the different ways a new functionality can be combined with existent ones. In a similar way, only have technical writers should be working with use cases - to properly describe the use case in an imperative way.

V. Conclusion & Future Work

Our study preliminary findings seems to reinforces the belief that a list of quality attributes can be used to guide the quality evaluation of BDD scenarios. However, our initial choice of attributes revealed that some may not be suited for BDD scenarios individually (like completeness or consistency) or may be only seen as a confusion source to the evaluator (like atomicity or independence). Therefore, the initial list of attributes that could work with BDD scenarios is: concise, estimable, feasible, negotiable, prioritized, small, testable, understandable, unambiguous, and valuable.

By analyzing the students evaluations of each other scenarios using a content analysis method [16], we can produce a list of synonyms for each quality attribute used on BDD scenarios. Those synonyms can be used on the next step of our research plan: to acquire feedback from industry practitioners and compare their evaluation criteria with the insights we got from our data - using those synonyms as a bridge between the attributes acquired on the empirical study and their opinions.

Another related next step is to ask those same practitioners if attributes lists are indeed a good way to validate BDD scenarios - the confusion on understanding the presence/absence of an attribute may have revealed the need to use a direct questionnaire, similar to Cockburn's one [5] for use cases.

References

[1] J. Smart, *BDD in Action: Behavior-Driven Development for the Whole Software Lifecycle*. Manning Publications, 2014.

[2] M. I. Kamata and T. Tamai, "How does requirements quality relate to project success or failure?," in *RE*, IEEE, 2015.

[3] S. Neely and S. Stolt, "Continuous delivery? easy! just change everything (well, maybe it is not that easy)," in *Agile Conference*, 2013.

[4] IIBA, *A Guide to the Business Analysis Body of Knowledge (BABOK Guide) 2nd Edition*. International Institute of Business Analysis, 2009.

[5] A. Cockburn, *Writing Effective Use Cases*. Addison-Wesley Longman Publishing Co., Inc., 2000.

[6] M. Cohn, *User Stories Applied: For Agile Software Development*. Addison Wesley Longman Publishing Co., Inc., 2004.

[7] G. Lucassen, F. Dalpiaz, J. VanDerWerf, and S. Brinkkemper, "Forging high-quality user stories: Towards a discipline for agile requirements," in *International Requirements Engineering Conference*, 2015.

[8] R. Jeffries, "Essential xp: Card, conversation, confirmation." http://ronjeffries.com/xprog/articles/expcardconversationconfirmation, 2001. Visited in: Jan. 2017.

[9] M. Gartner, *ATDD by Example: A Practical Guide to Acceptance Test-Driven Development*. Addison-Wesley Professional, 2012.

[10] V. T. Heikkila, D. Damian, C. Lassenius, and M. Paasivaara, "A mapping study on requirements engineering in agile software development," in *Euromicro Conference on Software Engineering and Advanced Applications*, 2015.

[11] IIBA, *A Guide to the Business Analysis Body of Knowledge (BABOK Guide) 3rd Edition*. International Institute of Business Analysis, 2015.

[12] P. Heck and A. Zaidman, "A quality framework for agile requirements: A practitioner's perspective," 2014.

[13] P. Heck and A. Zaidman, "Quality criteria for just-in-time requirements: just enough, just-in-time?," in *JITRE*, IEEE, 2015.

[14] K. Cox, A. Aurum, and R. Jeffery, "An experiment in inspecting the quality of use case descriptions," *Journal of Research and Practice in Information Technology*, 2004.

[15] R. Pichler, "The product vision board." http://www.romanpichler.com/blog/the-product-vision-board, 2011. Visited in: Jan. 2017.

[16] M. D. White and E. E. Marsh, "Content analysis: A flexible methodology," *Library Trends*, 2006.