Empirical Study to evaluate BDD scenarios quality

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. In this position paper, we argue the importance of what makes a "good" BDD scenario. Our intuition leads us to believe that the value of those documentation based scenarios is connected with how well they convey and document the details discussed by the team about the behaviors needed to fulfill customer needs. We take inspiration on the criteria used to evaluate other types of requirements (like use cases or user stories) to guide us on our reflection about how known quality attributes can be useful to BDD scenarios and in an on-going empirical study with novice BDD scenarios evaluators.**

***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 requirements, in the form of scenarios, together. Smart [1] states that those scenarios use a common language that allows for an easy, less ambiguous path from end-user requirements to usable, easy to automate scenarios. These scenarios specify how the software should behave and guide the developers in building a working software with features that really matter to the business.

However, there's a lack of studies who evaluates how those scenarios are written. It is well known that bad requirements are one of many potential causes of a project failure [2] [3] and that the complete, accurate and concise documenting of requirements is of vital, perhaps paramount importance within software development, and errors made in this phase are often considered the most difficult to solve and most costly to fix [4]. 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, thus bringing problems to teams like the one on Rally Software, reported by Neely and Stolt [5].

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. Therefore, we organized an empirical study with graduate students to understand how novice evaluators use know quality criteria to judge the quality of BDD scenarios.

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 our impressions during this study. Section 4 concludes this paper by summarizing our perceptions about the on-going study and outlining some directions for future research.

II. Background

A. Requirements

A requirement is either a condition or capacity necessary to solve a problem or reach a goal for an interested party or some characteristic that a solution or component should possess or acquire in order to fulfill some form of contract [6]. When classified according to their purpose, they can be called business requirements, stakeholder requirements and solution requirements. In this paper, we focus on the later, solution requirements, which describes the capabilities and qualities of a solution and provide the appropriate level of details to allow the proper implementation of that solution. More precisely, we focus on a particular type, called functional requirements, that describes the capabilities a solution must have in terms of the behavior and information to manage.

B. Use Cases

Cockburn [7] 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 want to perform an action and achieve a certain goal). Different sequences of behavior, or scenarios, can unfold, and the use case collects together those different scenarios. They are used to express behavioral 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 has other parts as well: 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* is a case in which nothing goes wrong and the *extensions section* describes what can happen differently during that scenario.

C. Behavior-Driven Development

Most agile methodologies tend to not use traditional requirements or use cases, but represents requirements using user stories. For Cohn [8], a user story describes functionality that will be valuable to either a user or purchaser of a system or software. Lucassen et. al [9] summarize that user stories only

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

According to Jeffries [10] an user story is composed of three elements. The Card (the expression of the essential elements of a requirement) represents customer requirements rather than document them - therefore, it has just enough textual information to identify the requirement and remind everyone what the story is about. The Conversation is an exchange of thoughts, opinions, and feelings. It is largely verbal but can be supplemented with documents. The best supplements are examples and the best examples should be executable. They are representations of the Confirmation, a way to customers tell developers how she will confirm that they have done what is needed in the form of acceptance tests. 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 settle what needs to be done and write those details in the format of acceptance tests.

Behavior-Driven Development 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 pre-condition *When* an action is performed *Then...* a result is observed. For different actions, different scenarios should be created, an approach fulfill the purpose of the use case's extension section.

It's a known fact that BDD scenario is a format to represent acceptance tests [11], as it fulfills the role of the Confirmation term defined by Jeffries [10] by specifying scenarios who convey the product behavior. The use of acceptance tests as documentation is highlighted by Neely and Stolt [5] on their experience at Rally Software. When planning stories that require modification of existing code, they state that the lengths of documenting existing tests can be used as guidance when discussing the story details. It helps to enhance the team's visibility about code's test coverage and also allows QA and developers to close existing gaps, boosting the team level of confidence on that part of the application.

D. 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 [12].

The Business Analyst Body of Knowledge (BABOK) newest edition [13] 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 [6], as follows: cohesion, completeness, consistency, correction, viability, adaptability, unambiguity, and testability. Although the characteristics' meaning is defined, no measurement guidance is given. Cockburn [7] take 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.

Heck and Zaidman [14] [15] have not found a practical implementation to perform the verification for agile requirements quality. Even though recognize existing approaches, such as the heuristics of the INVEST (Independent-Negotiable-Valuable-Estimable-Scalable-Testable) framework described by Cohn [8], their assumption that the notion of quality for agile requirements is different from the notion of quality for traditional upfront requirements led them to create a framework for quality criteria for agile or open source requirements. Lucassen et. al [9] define additional criteria on top of Heck and Zaidman [15] to evaluate user stories on their QUS Framework.

However, little attention is being given to the quality of that written documentation on BDD scenarios format. Using Jeffreys [10] terms, we believe that, if the Confirmation is not a good representative of the details discussed in Conversations by the team and the customer, the simple approach of writing customer needs on Cards is not effective. To the best of our knowledge, the only guide practitioners have to validate their scenarios are taken from tips based on few examples described by Smart [1] in his book, as follows: the 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 an 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. The author specifies examples of good and bad scenarios in order to demonstrate those characteristics.

III. Evaluating BDD Scenarios Quality

Before creating a quality questionnaire similar to the one used on Cockburn's use cases [7] or the refined quality rules proposed by Phalp et. al [4], we must understand what criteria are important to evaluate the quality of BDD scenarios. Therefore, we must list the quality attributes that would work with BDD scenarios, in a similar way that traditional attributes [6] [13] work with requirements documents and INVEST heuristic [8] works with user-stories. In preparation to creating that listing, we first seek to gain insight on whether the existing quality attributes would work with BDD scenarios, and if they do not, which ones would be a good fit for such purposes.

For that reason, we asked graduate students to use traditional attributes from both BABOK editions [6] [13] and the INVEST framework [8] to evaluate BDD scenarios. Our goal was to understand how those attributes would be used to evaluate BDD scenarios in order to bring to light the most suited attributes to evaluate BDD scenarios, as similar studies executed in the past did with Use Cases [7] [16] [17]. 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.

However, we had two concerns before starting: (a) we were not certain about how a person's evaluation of a textual document using quality attributes would differ based on the representation format and (b) we wanted to have evaluators who understood the product enough to judge its requirements - therefore, we could not ask students to evaluate requirements for a product domain they have no understanding of.

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 formats seems to prefer a more declarative way to describe a behavior, rather than imperative, to stay detached from the implementation details [7] [1]. Also a use case execution path seems to be a BDD scenario on a different format - on Cockburn's book [7], the main scenario is always described.

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

Therefore, we asked the students to perform the following sequential tasks:

1) Develop functional requirements for product A with requirement format assigned;
2) Develop functional requirements for product B with another requirement format assigned;
3) Evaluate other student functional requirements for product A;
4) Evaluate other student (different from the item above) functional requirements for product B;

Each student was assigned a requirements format to perform the writing tasks (tasks 1 and 2). They had to use either user stories and BDD scenarios (US + BDD) or use cases (UC). Therefore, we had virtually two group os students - Group 1 (G1) contained those who develop BDD scenarios for product A and use cases for product B and Group 2 (G2) those who develop use cases for product A and BDD scenarios for product B. For the evaluation tasks, students on one group should receive functional requirements from the other group. Thus, students from G1 who developed BDD scenarios for product A should now evaluate the use cases for product A as part of task 3. Consequently, students from G2 who developed use cases for product A should now evaluate BDD scenarios 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.

The products were presented to the students in a product vision board format [18]. On that occasion, they had the chance to ask questions, straighten their understanding on how to solve the business problem presented, and collaboratively draft some high-level features. After this initial discussion, we would take their suggestions and create the final high-level features. Finally, a new discussion round was performed so the students could read the features and had the chance to validate them, negotiate their details, decide whether each feature should be part of the first release of the product or not. Those discussion sessions were planned to give them all the same understanding of what the product should be and how it should behave, as a means to guarantee a common ground for the students to produced use cases, stories, and BDD scenarios. After the last round of discussions, the students then proceeded to perform tasks 1 to 4 in an 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 directly to them. Those social network interactions would provide resellers with enough information to direct their marketing efforts and promotions to the right subset of users in an special website area where one could gather and visualize user data.

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.

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)*

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.

First of all, 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 [6] [13] and the INVEST heuristic provided by Cohn [8], some students did not understood how to proper differentiate some attributes when evaluating scenarios. For example, atomicity from BABOK [13] and independence from INVEST [8] 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.

Secondly, some attributes may not make sense to evaluate a single BDD scenario - completeness may 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. Thus, five students out of the 15 have joined the scenarios from a user story together before analyzing them, witch raises questions on what were the reasons that drove their decision. Follow up sessions need to be scheduled with those students to search for the reasoning behind their evaluation approach.

Thirdly, the students reported that inputs/outputs are important on BDD scenarios to help on prioritization and testability. As they don't have use cases implementation details to help them understand the technical steps needed to perform an action, stating inputs and outputs clearly may impact the a development team estimation effort and impact analyzing, as well as helping to better evaluate a scenario testability. A few of them understands that the lack of inputs/outputs can be seen as a lack of the completeness attribute. Those perceptions are similar to the good examples given by Smart [1] on his book - however, the lack of input/output is not explicitly provided by him as a good/bad thing.

Fourthly, written rigor was judged as necessary by the students to re-enforce the team's common understanding during conversations with customers. As they stated, the writing of BDD scenarios is easier (due to the use of plain English language to represent behaviors) and could theoretically be done by anyone - however, they reported that someone who provides good examples is desirable, as to better demonstrate to the development team the different ways a new functionality can be combined with existent ones.

V. Conclusion & Future Work

Our study seems to reinforces the belief that a list of quality attributes is important to guide the quality evaluation of BDD scenarios. However, that initial list of attributes revealed that some may not be suited for BDD scenarios individually (like completeness) or may be only seen as a confusion source to the evaluator (like atomicity or independence). By analyzing the students evaluations of each other scenarios using a content analysis method [19], we came up with a list of synonyms for each quality attribute used on BDD scenarios - for example, for non-ambiguity, we came up we words like "direct" or "clear" while for prioritized we got "importance" or "ranking". Our next step is to acquire practitioners understandings about what makes good BDD scenarios and cross their opinions with those synonyms, to map them back to existing attributes or provide us with words clusters to use as new ones.

One related next step is to verify if attributes lists are indeed a good way to validate BDD scenarios - the confusion on understand the presence/abscence of an attribute showed by students may reveal the need to evaluate scenarios with direct questions, as those developed to use cases by Cockburn [7].

Therefore, our next steps to achieve our goal are to better understand how acceptance tests are evaluated, what problems may appear during their creation or use on a continuous engineering context and map those problems on BDD scenarios. This knowledge will give us confidence to build a question based checklist to avoid those problems.

References

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

[2] The standish group, "CHAOS." https://www.projectsmart.co.uk/white-papers/chaos-report.pdf, 2015. Visited in: Jan. 2017.

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

[4] K. Phalp, A. Adlem, S. Jeary, J. Vincent, and J. M. Kanyaru, "The role of comprehension in requirements and implications for use case descriptions.," *Software Quality Journal*, 2011.

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

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

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

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

[9] 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.

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

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

[12] 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.

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

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

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

[16] 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.

[17] S. Tiwari and A. Gupta, "A systematic literature review of use case specifications research," *Information and Software Technology*, 2015.

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

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