How to Evaluate BDD Scenarios' Quality?

Gabriel Oliveira
gabriel.pimentel@acad.pucrs.com
MuDDOs Research Group - PPGCC
School of Technology, PUCRS
Porto Alegre, RS, Brazil

Sabrina Marczak
sabrina.marczak@pucrs.br
MuDDOs Research Group - PPGCC
School of Technology, PUCRS
Porto Alegre, RS, Brazil

Cassiano Moralles
cassiano.moralles@edu.pucrs.br
MuDDOs Research Group - PPGCC
School of Technology, PUCRS
Porto Alegre, RS, Brazil

ABSTRACT

A scenario from the Behavior-driven development (BDD) practice is a known format to represent acceptance tests in agile methodologies, communicating assumptions and expectations by expressing the details that result from the conversations between customers and developers. We believe that this formalization of behavior need to be of good quality to avoid known requirement problems that arise from bad documentation, such as incomplete, underspecified and inconsistent requirements. However, there are only informal guidelines to guide practitioners on their BDD scenarios' elaboration and quality evaluations. To address this lack of guidance, we define a set of quality attributes and propose a question-based checklist to assist BDD scenarios' quality evaluations. [Methods] The quality attributes were identified from an interview-based study with 18 practitioners. In this study, practitioners shared their interpretations on an initial set of literature-informed quality attributes and their own personal evaluation criteria. We consolidated both in a single list of newly redefined attributes, used in the definition of our proposed checklist. We believe that our newly re-defined quality attributes and question-based checklist can enhance the existing guidelines and practitioners' ability to evaluate BDD scenario's quality by providing them with an standard guideline for scenarios' refinement conversations.

CCS CONCEPTS

• **Software and its engineering** → Software development techniques.

KEYWORDS

behavior-driven development, quality, documentation, checklist, empirical study

ACM Reference Format:
Gabriel Oliveira, Sabrina Marczak, and Cassiano Moralles. 2019. How to Evaluate BDD Scenarios' Quality?. In *XXXIII Brazilian Symposium on Software Engineering (SBES 2019), September 23–27, 2019, Salvador, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3350768.3351301

SBES 2019, September 23–27, 2019, Salvador, Brazil

ACM ISBN 978-1-4503-7651-8/19/09…$15.00
https://doi.org/10.1145/3350768.3351301

1 INTRODUCTION

Behavior-Driven Development (BDD) is an umbrella term to describe a set of practices that uses scenarios as an ubiquitous language to describe and model a system [13]. BDD scenarios, a known format of acceptance tests [6], are expressed in a format designed to be both easily understandable by business stakeholders and easy to automate using dedicated tools [13]. Smart [13] states that bringing business and technical parties together to talk about the same document helps to build the right software (the one that meets customer needs), a thought reinforced by Wynne and Hellesoy [16].

Wynne and Hellesoy [16] understand that many software projects suffer from low-quality communication between domain experts and programmers on a team, a known requirements engineering problem [4]. BDD scenarios help to avoid this problem by building scenarios in a common language that allows for an easy, less ambiguous path from end-user business requirements to scenarios that specify how the software should behave and guide the developers in building a working software with features that really matter to the business and its customers and end-users [13].

To the best of our knowledge, writers of BDD scenarios acting on software development teams do not have a standard set of rules to educate themselves on what a "good" BDD scenario is. They can only compare their work with a few guidelines and examples of "good" and "bad" scenarios found on Smart's book [13], Wynne and Hellesoy's book [16], and other informal internet references. This comparison can be misguided as the writer's application context may not be comparable to the books' examples context.

Due to that fact, we fear that BDD scenarios mitigation of communication problems, during the discovery and definition of features [14], may be lost due to the unguided formalization of those features in the form of BDD scenarios, which may lead to other known requirement problems such as incomplete, underspecified and inconsistent requirements [4].

We believe that structuring the tacit knowledge of BDD practitioners could enhance the already existing guidelines from other sources and similar purposes (e.g., [13][16]), and the practitioners' ability to evaluate their own BDD scenarios' quality by providing them with an standard guideline that can be the input of scenarios' refinement conversations. Therefore, the two-fold goal of this paper is to structure that knowledge in the form of quality attributes and to propose a question-based checklist, a document accessible to software teams [17], similar to Cockburn's on use cases [1].

To that end, we conducted semi-structured interviews with 18 practitioners. As a resource to stimulate practitioners to think about how they define quality for their BDD scenarios, those interviews used literature-informed quality attributes from traditional requirements [10][9] and user stories [2]. Practitioners' interpretations of

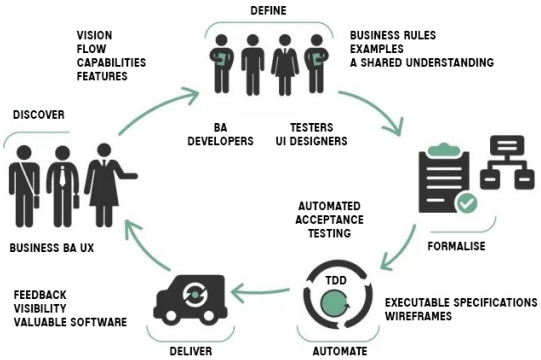

Figure 1: Typical stages of a feature on a BDD process [14]

```
@javascript
Feature: Blocking a user from the stream
    Background:
        Given following users exist:
            | username    | email             |
            | Bob Jones   | bob@bob.bob       |
            | Alice Smith | alice@alice.alice |
        And a user with email "bob@bob.bob" is connected with "alice@alice.alice"
        And Alice has a post mentioning Bob
        And "alice@alice.alice" has a public post with text "All your base are belong to us!"
        And I sing in as "bob@bob.bob"

    Scenario: Blocking a user
        When I confirm the alert after I click on the first block button
        And I go to the home page
        Then I should not see any posts in my stream

    Scenario: Blocking a user from the profile page
        When I am on "alice@alice.alice"'s page
        And I confirm the alert after I click on the profile block button
        Then I should see "Stop ignoring" within "#unblock_user_button"
        And "All your base are belong to us!" should be post 1
        When I go to the home page
        Then I should not see any posts in my stream
```

Figure 2: Blocking a user feature file from Diaspora [5]

the literature-informed quality attributes and their own personal evaluation criteria were consolidated to form 8 newly redefined quality attributes. From those, we derived our question-based checklist, which use is demonstrated through a proof of concept exercise.

This paper proceeds as follows: Section 2 reviews the concept of BDD scenarios and how they are created. Section 3 reflects upon the different set of criteria to validate other requirements. Section 4 presents the research design we followed to develop the understanding of which quality attributes are used and how these are used in BDD scenarios' evaluations. Section 5 presents the interview findings, summarizing the participants' interpretations of the literature-informed quality attributes and their personal evaluation criteria. Section 6, this paper's main contribution, groups those interpretations and personal criteria into newly redefined quality attributes and presents a proposed question-based checklist to evaluate BDD scenarios' quality. Section 7 concludes the paper and outlines future research.

2 BACKGROUND

Agile software development methods tend to be guided by conversations about requirements rather than formal documents and defined phases from traditional requirement engineering approaches [8]. Most of those methods employ user stories, which describe functionality that will be valuable to either a user or purchaser of a system or software [2]. Lucassen et. al [11] 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. To complement user stories, acceptance tests help the customer to communicate assumptions and expectations to the developers. These tests express the details from conversations between customers and developers, while validating that a story has been developed with the functionality the customer and the team had in mind when they wrote it [2].

A known format of acceptance tests is BDD scenarios. Behavior-Driven Development is a set of practices that uses scenarios as an ubiquitous language to describe and model a system [13]. Smart [14] describes each stage of a feature worked by a team following a BDD process as a set of cyclical stages as shown in Figure 1. During the *discover* stage, high level techniques help to get an overall picture of what one is trying to achieve, generating stories that will be detailed later, once needed. In the *define* stage, the team starts to have concrete conversations around more specific business rules and examples of how the user would interact with the system. In the *formalize* stage, where our study is focused on, key business rules and examples are formalized into scenarios.

The scenarios related to a particular feature are grouped into a single text file called a feature file written in the Gherkin language, like the one in Figure 2. A feature file contains a short description of the feature, followed by a number of scenarios, or formalized examples of how a feature works. Each scenario is made up of a number of steps, where each step starts with one of a small set of pre-defined keywords. The natural order of a scenario is *Given*... a context *When*... an action is performed *Then*... an outcome is obtained. BDD scenarios are similar to use cases scenarios as both describe a system behavior under certain precondition (expressed on the *Given* clauses of a BDD scenario) to achieve a certain goal (expressed on the *Then* clauses of a BDD scenario).

3 RELATED WORK

The BABOK's 3rd edition [10] argues that one way to validate quality characteristics is through a review. Zhu [17] states that software review is primarily an individual effort and the types of reading techniques an individual uses are paramount to the outcome and effectiveness of software review. The most widely used format of reading technique is checklists [17]. Checklists can be represented as a list of questions to provide reviewers with hints and recommendations for finding defects during the examination of software artifacts. One example of a question-based checklist is the one used by Cockburn [1] to evaluate the quality of Use Cases.

Additionally, the BABOK's 2nd edition [9] describes eight characteristics a requirement must have in order to be a quality one, as follows: adaptability, cohesion, completeness, consistency, correction, testability, unambiguity, and viability. BABOK's 3rd edition [10] brings nine: atomic, complete, consistent, concise, feasible, prioritized, testable, unambiguous, and understandable. Both editions [10][9] define what each characteristic means, but do not provide any guidance on how to use them when evaluating a requirement.

For User Stories, a format to represent agile requirements, the INVEST (Independent, Negotiable, Valuable, Estimable, Scalable, Testable) acronym presented by Cohn [2] seems to be one of the mostly used criteria, as identified in Heck and Zaidman's empirical study [7]. The authors have used INVEST attributes as the criteria

to evaluate user stories in their framework to evaluate just-in-time requirements's quality [7], along with other criteria such as: basic elements as role, activity, and business value; acceptance criteria or acceptance tests to verify the story; uniformity, forcing each user story description to follow the standard user voice form; and attachments represented in a uniform modelling language.

Also, Lucassen et. al [11] define additional criteria to evaluate user stories on their QUS Framework, as follows: atomic, minimal, well-formed, conflict-free, conceptually sound, problem-oriented, unambiguous, complete, explicit dependencies, full sentence, independent, scalable, uniform, and unique.

To the best of our knowledge, BDD scenarios can only be evaluated based on characteristics taken from Smart [13] and Wynne and Hellesoy [16] experiences, such as: scenarios steps expressiveness; focus the steps 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 these authors [13][16] specify a few scenarios as examples to demonstrate those described characteristics, BDD scenarios could benefit from a question-based checklist defined from the collective knowledge of other practitioners, similar to the one used on Cockburn's use cases [1].

4 RESEARCH METHODOLOGY

BDD scenarios represent behaviors that are taken from conversations within a team, in a similar way that use cases are created. Therefore, it would make sense to start evaluating BDD scenarios with known quality attributes already used in traditional requirements. Also, as BDD scenarios complement user stories, one could infer it would be safe to use some of those attributes [2] to evaluate both. However, a pilot study [12] has shown that some attributes may be only seen as a confusion source to the evaluator. We believe this confusion came from the many ways to interpret the meaning of a quality attribute, such as concise. In BDD scenarios, concise can mean a scenario has a few steps, or that a step has no unnecessary information, or, yet, that each step is focused in describing a single thing. We choose to consider all different interpretations valid and use those literature-informed quality attributes only as a support for the semi-structured interviews, allowing practitioners to interpret these attributes using their own tacit knowledge evaluating real-world BDD scenarios. Those different interpretations and practitioners' personal quality criteria were grouped together into newly redefined quality attributes that are the foundation of the question-based checklist we propose later in this paper, which provides practitioners with an standard guideline that can be the used on scenarios' refinement conversations.

4.1 Research Goal and Questions

The main goal of our research, presented in this paper, is two-folded: to identify quality attributes based on the knowledge of BDD practitioners and to propose a question-based checklist to evaluate BDD scenarios. To accomplish that, two research questions were posed to drive us forward. Research Question 1 (RQ1) asks *"What are the quality attributes suited to describe a good BDD scenario by the view of a software development team member?"* and is answered by the newly redefined quality attributes identified in the interview-based empirical study reported in Section 6.1. Research Question 2 (RQ2) asks *"How does a software development team member evaluates BDD scenarios with those attributes?"*. This question is answered by our proposal of a question-based checklist to assess BDD scenarios' quality, reported in Section 6.2.

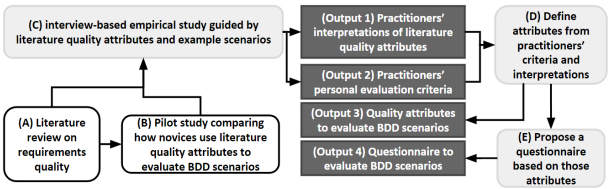

Figure 3: Research Design

4.2 Research Design

To achieve our research goal, we proposed a multiple-steps research design as presented in Figure 3, where the light gray boxes are the actions reported in this paper that achieved the outputs shown in the dark gray boxes. This design was based upon the understanding that interviews with open-ended questions alone would not provide enough data to formulate a proper question-based checklist due to the participants' plurality of terms and opinions. Therefore, we judged necessary to have our interviews guided by a set of quality attributes that had previously been used to evaluate BDD scenarios. A literature review (Figure 3, Step A) has confirmed that traditional attributes [10][9] and the INVEST [2] acronym were used with agile requirements. A pilot study with 15 graduate students [12] (Figure 3, Step B) identified how effective those attributes could be when used with BDD scenarios. The output of that study was a subset of the previously identified quality attributes from literature, as follows: concise, estimable, feasible, negotiable, prioritized, small, testable, understandable, unambiguous, and valuable.

This paper reports on this semi-structured interview-based empirical study (Figure 3, Step C) and onward (Steps D and E). In this study, in addition to the literature-informed attributes, we used known examples of BDD scenarios to aid practitioners realize their own quality criteria while reviewing those scenarios. To avoid our own bias towards what would be a good or bad BDD scenario, we decided to not create the examples ourselves. Instead, we handed them real scenarios from the Diaspora [5] open source project, a decentralized social network that employ BDD scenarios to detail the application behavior. To the best of our knowledge, Diaspora is the open source project with the most feature files available.

The interview-based study resulted in a list of practitioners' interpretations of literature-informed quality attributes (Output 1, reported in Section 5.1) and their own personal criteria (Output 2, equally reported in Section 5.2). With that data at hand, we consolidated these characteristics into a single list of newly-redefined attributes to evaluate BDD scenarios (Output 3, see Section 6.1) used to define our question-based checklist and perform a proof of concept on it (Output 4, refer to Section 6.2).

Table 1: Participants Profiles

Participant	Role	BDD experience	Location
P1	Tester	3 years	England
P2	Tester	3 years	Netherlands
P3	Developer	3 years	Netherlands
P4	Coach	3 months	Denmark
P5	Tester	9 months	Netherlands
P6	Tester	2 months	Netherlands
P7	Tester	3 years	Netherlands
P8	Dev/Coach	10 years	Hungary
P9	Coach	3 years	England
P10	Developer	6 years	Sweden
P11	Tester	4 years	Australia
P12	Tester	3 years	Brazil
P13	Developer	3 years	Brazil
P14	Tester	1 year	Brazil
P15	Coach	5 years	Australia
P16	Developer	1 year	Brazil
P17	Tester	4 years	United States
P18	Tester	4 years	England

4.3 Interview's Participants Selection

In order to identify the first participants for our interviews, we organized direct searches on our own Linkedin social-network using the terms "BDD" and "Behavior-Driven Development". Our belief was that a personal touch, showing our profiles and highlighting practical experience in industry to potential participants, would improve our chances of finding good candidates. Within every practitioner profile, we double-checked the existence of BDD experience and proceeded to invite people to connect and dedicate their time for a one-hour long interview. Eighteen people accepted to be interviewed. The participants profiles are summarized in Table 1.

Each interview lasted in average 77 minutes, being the longest 1 hour and 51 minutes and the shortest 62 minutes long. A total of 1390 minutes were recorded. The interviews were conducted via Skype and voice-recorded with permission. The interviews were performed either in English or Portuguese during a 3 months period (Sept to Nov 2017) and then manually transcribed to the participant's native language. After each interview, the first author would send a summary of the findings and highlights to the second author, and within 24 hours they would discuss the results and assess whether news findings were still coming along. This debriefing process allowed us to decide when to stop conducting more interviews for identifying theoretical saturation [15].

4.4 Interview Design

Taking inspiration on Heck and Zaidman's interview questions [7] with practitioners to understand how user stories' quality can be understood, we list in Table 2 the set of questions we used in our study to capture practitioners' interpretations of literature-informed quality attributes and their own evaluation criteria. Heck and Zaidman's [7] original interview script was organized into two parts: one done with minimal introduction from their side and without showing the participants their framework, and another in which they asked the participants to use their framework transformed into a checklist on some examples taken from open source projects. In the same way, in our interviews, we keep our list of literature-informed quality attributes to ourselves up to Question 7 so the participant could discuss her own list of quality criteria when asked to review a Diaspora's feature file of her choice. We then move to the second half of the interview (Questions 7-13), where we explicitly present the literature-informed quality attributes to the participant without defining their meaning. With that list in hand, we ask the participant to share her interpretation of each one of the attributes, to use them into another Diaspora's feature file of her choice, to consider how suited the literature-informed attributes are to evaluate BDD scenarios, and to map them to their own criteria – making sure no criteria or attribute is forgotten.

Table 2: Interview Questions

ID	Question
1	What is your role on the project?
2	What is your main task on the project?
3	For how long do you use BDD?
4	How does your project use BDD scenarios?
5	What do you pay attention to when evaluating BDD scenarios?
6	On Diaspora, evaluate a feature file according to your criteria.
7	Do quality attributes help evaluating BDD scenarios?
8	What is the meaning of each attribute on BDD scenarios?
9	On Diaspora, evaluate a feature file according to the attributes.
10	Do you miss any other attribute?
11	What quality attributes did you find difficult to use?
12	To what extent the attributes helped you evaluate a scenario?
13	How do your criteria map to those attributes?

More specifically, Questions 1 to 4 were used to understand the participants profile as summarized in Table 1. Questions 5 and 6 were useful to identify the participant's criteria without any bias of our literature-informed attributes. These are presented in Section 5.2. If only a few answers emerged for Questions 5 and 6, we used the criteria from Smart [13] and Wynne and Hellesoy [16] experiences to provoke the participant's thoughts. Questions 7 to 11 were useful to identify the participant's interpretations of our literature-informed quality attributes. Results are reported in Section 5.1. Question 12 was useful to assess how useful such a list was and tease the participant to suggest better ways to evaluate BDD scenarios. Question 13 was a final check in case the conversation have yet not linked participant's criteria and interpretations to attributes. We present the results from the second part of the interview first to increase the comprehensibility of our findings.

Taking the RQ lens, Questions 5 and 8 helped reveal the meaning practitioners see on quality attributes and on their own criteria, thus allowing the newly defined quality attributes that answer our RQ1 to emerge. In addition, Questions 6 and 9 gave us insights on

how those attributes and personal criteria were used in practice, providing important information to answer RQ2 and define our question-based checklist.

Finally, Question 13 asked the participant to link those literature quality attributes with their own criteria, tying those attributes with practical details of BDD scenarios and motivating us to threat interpretations and criteria altogether in our checklist. However, our initial thought that this mapping would be beneficial was not successful, due to the different interpretations of each attribute. As we could not decide for ourselves what interpretation of each attribute is more suited than others without enforcing our own bias, we used them all as characteristics, regardless of what literature attribute generated it. Additionally, the participants' personal evaluation criteria were also treated as characteristics. Those two sets of characteristics were later grouped together into newly-redefined quality attributes, that compose our proposed question-based checklist as presented in the following sections.

Before use, the script was validated by a doctoral student with 4 years of previous experience in industry. Additionally, a pilot interview was conducted with a software tester who has about 6 years of experience with BDD usage. He reinforced our idea of using Smart [13] and Wynne and Hellesoy [16] experiences to provoke the discussions in Questions 5 and 6.

4.5 Data Analysis

Cruzes and Dyba state [3] that a number of different methods have been proposed for the synthesis of qualitative and mixed-methods studies such as the ones we typically find in Software Engineering (SE). This lead them to conceptualize thematic synthesis in SE. Thematic synthesis draws on the principles of thematic analysis to identify the recurring themes or issues in the primary sources of data, analyzes these themes, and draws conclusions from them.

Knowing that we specifically decided to guide our interviews with literature-informed quality attributes (Questions 7 to 11) and with certain criteria inquiries (Questions 5 and 6) from Smart [13] and Wynne and Hellesoy [16] when necessary, we used those as initial thematic analysis codes. We refined these codes along into themes during our cyclical analysis, resulting in the newly-redefined quality attributes that compose our proposed question-based checklist as presented in the following sections. The final codes that compose those newly-redefined quality attributes are presented in Section 6.1.

4.6 Limitations

Only the main researcher was involved into the coding process, which may have impacted the themes creation in an unforeseen way, even with the careful review of the second researcher. Additionally, we have not taken into consideration the gender, role or location, nor the type of industry a participant works to reflect upon the data taken from the interviews. We understand that different life experiences may have brought different quality criteria and opinions, so we tried to mitigate this effect interviewing practitioners from different areas and companies. The choice of using BDD scenarios from the Diaspora open source real-life project might also be a threat to the interview results. Choosing different sets of BDD scenarios might have yield different results than ours. We discussed this matter during the interview validation and pilot interview, and both professionals judged Diaspora's feature files as good representatives of real-world BDD scenarios. Finally, we could have phrased the items in our question-based checklist in many different ways, so we have to acknowledge that our own language bias may have driven us to write them in that way we presented. The used wording might impair the checklist applicability. To this end, a validation of our checklist by other practitioners is listed as a future work to refine the artifact and avoid this threat. For now, we demonstrate its applicability with a simple proof of concept.

5 RESULTS

This section presents the participants (from P1 to P18) responses using the examples from the Diaspora project and the set of literature-informed quality attributes resulting from our pilot study. Recalling, these attributes are: concise, estimable, feasible, negotiable, prioritized, small, testable, understandable, unambiguous, and valuable. Our interview-based study (Figure 3, Step C) goal was to identify the participants' interpretations of these attributes (Figure 3, Output 1) and how participants' own personal criteria are linked to these attributes (Figure 3, Output 2).

5.1 Literature-Informed Quality Attributes' Interpretations

Participants' interpretations of each literature-informed quality attribute came from the answers of Questions 7 to 10 in our interview script (see Table 2).

For instance, concise attribute was deemed important by all 18 participants. According to them, a concise BDD scenario has no unnecessary details (P3, P5, P8, P10, P11, P16, P17), is focused on the problem and not on the technical solution (P4, P7, P10, P12, P13, P14, P18), is clear (P1, P3), and has only a few (P2, P7), brief and comprehensive steps (P3, P6, P9, P11, P18). P5 reports that a concise scenario *should be specific without giving way too much detail* and should have *no steps longer than needed*. P14, who interpreted concise as focused, said a scenario should be *straight to the point, without unnecessary details* and that *it does not need to be very long, trying to explain point-by-point, it can be more direct*, an interpretation followed by a rewriting suggestion of a scenario to a declarative writing format rather than an imperative one.

A small scenario, often considered the same as a concise scenario (P2, P7, P8, P9, P15), is one with just a few steps (P1, P3, P4, P10, P11, P13, P14), which test only one thing (P5, P6). P10 expands on his opinions by saying that *if I need a lot of words to express an specific example, it's certainly not small. And if one needs lots of lines to do the same thing, it [the scenario] is probably not small too.*

A testable scenario is one which allows the reader to follow its steps (P1, P12, P14, P18), has clear outcomes (P2, P4, P9, P15) and pre-conditions (P6), is focused on the problem and not on the technical solution (P5), and covers all the aspects of the feature (P12) when considering the scenarios together. For P1, a testable scenario means that *it should be clear and be something easy to understand so you can follow the steps.* For P2, testable means that *scenario's intended behavior, or what you are trying to verify, should be clearly expressed.*

For some participants, an understandable scenario uses an ubiquitous language (P2, P3, P5, P6, P8, P9, P10, P11, P15), is self-contained (P18), and is written in "good english" (P17). P11 defines understandable as ubiquitous, and define it as *a terminology that both your technical and business people understand and it is specific to the software business domain.*

Ambiguity on scenarios was interpreted as the use of vague statements, weak words and contradictions (P2, P3, P4, P5, P12, P14), the lack of a single scenario's intention (P7, P11, P15, P17), the fact that scenarios with different descriptions test the same thing (P1, P3), the lack of enough test coverage (P6), and steps' descriptions with high granularity (P9, P16). One of the sources of ambiguity is bad use of language by using vague, not clear statements such as the steps *Then the outcome is ok* or *Then the result is good* (both suggested by P2) or *When we see the change in the GUI* (suggested by P4), which gives room for different interpretations (P5, P14). Also, P12 raised the concern of having one step contradicting the other. Finally, multiple *When* steps (P7, P11) causes the scenario's intention to not be clearly stated.

A valuable scenario is unique (P6, P10, P11, P15, P18) by validating different and interesting behavior. Some participants could not identify how valuable a scenario is by reading the scenario' description alone. For them, how valuable a scenario is would have already been discussed by other team members during formal or informal conversations (P5, P8, P9, P17) related to the define stage in Figure 1. Some participants preferred to say a scenario should have value for the business (P1, P12, P13, P14), to the technical team (P16), as a documentation source (P3), or as a communication tool (P2). P6 reflects upon the term valuable by asking *is this scenario going to add value to whatever we are doing?*. P11 agrees that uniqueness should belong to valuable, along with the how important the scenario is for business people. P8 understands that valuable comes from discussions with the team as *we only define scenarios for things that are valuable.*

Prioritized, negotiable, feasible, and estimable attributes were largely regarded by our participants as not useful to evaluate BDD scenarios textual descriptions, as either they are useful only for conversations around scenarios, or dependent on steps' technical implementation knowledge or project's domain knowledge. P11, who declared those attributes as useful for conversations only, explains his thoughts on those attributes: *So I think the other [attributes] would already have been considered by the time you reach that [formalization] stage, because, if you remember, I was talking about example mapping, which is an activity that comes before writing feature files. So, when you write your examples you need to make sure they are estimable, for example. And obviously your scenarios would be derived from examples and, therefore, will automatically be estimable. So it's more relevant at the time of example mapping [define stage] rather than feature file formalization.* Similar declarations were heard from the participants who either have conversations before the writing of scenarios or who validate them after they are written.

Some characteristics that were not tackled by our list of literature-informed attributes were quoted as important as well, like completeness (P1, P4, P11, P18), declarative rather than imperative writing format (P10, P11, P18), cohesion (P12, P16), integrity (P17), and consistent writing patterns (P3, P8). Completeness was already mentioned as part of testable, as well as the use of declarative description rather then imperative was part of the focused interpretation for concise attribute. Cohesion, for P16, means that *scenarios should have a reason to belong to a given feature file*, which matches the sense of uniqueness of a scenario already found on the valuable attribute. The integrity of a scenario was regarded by P17 as the misuse of the BDD keywords when he declared that *I have sometimes seen people try to circumvent the rule of strict steps ordering. They start turning Then verifications into When steps, they will say something like "When I'm on Alice's page, And I should look at this dog, Then I focus And ..." – they are violating the integrity of the steps type, and that's no more behavior driven than the original procedure*–an aspect that did not match anything else we have heard. Writing patterns could be translated as the need for consistency of the use of business terms, similar to the need of actor consistency mentioned as a language criteria on the next section.

5.2 Participants' Personal Criteria

Participants' personal evaluation criteria, taken from the answers of Questions 5 and 6, revealed us a number of good and bad practices related to literature-informed quality attribute. To aid on their reporting, the identified criteria have been organized into four groups: language criteria, steps criteria, title criteria, and others.

Language characteristics depend on the consistent use of business terms instead of technical ones, the declaration of actions rather than the description of steps, and the writing from an actor point of view. More specifically, participants have mapped the consistent use of business terms, like *a glossary that appears consistently on all scenarios* (P9), as a practice positively impacting the literature-informed understandable (P1, P2, P3, P5, P6, P7, P8, P9, P10, P11, P15, P18), unambiguous (P3, P8, P10, P13), and concise (P4, P11) attributes and as a pain-point for the estimable (P3) attribute – for P3, *if it's on technical language it's very easy to estimate.* In opposition to the use of business terms is the use of technical language, considered as harmful for the understandable (P1, P2, P5, P6, P8, P9, P11, P14, P17, P18), concise (P2, P10), unambiguous (P2), small (P10), valuable (P7), and testable (P10) attributes. Examples of technical language, that should be avoided, would be the use of HTTP response codes (P5) like *Then the response is 200 OK*, or elements of the user interface like *When I click on a button.*

There is also the concern about how granular a scenario step description should be. P2 says scenarios should *focus on the problem, not on the technical solution* and P8 calls it as *declarative*, with *imperative* being his opposite. For P17, *Gherkin is meant to be declarative because it tries to describe behavior that add business value, it's not necessarily to define the implementation on how that behavior works.* For those participants who mix scenarios' writing with conversations the use of declarative language is seen as a good practice that enhances understandable (P7, P8, P9, P14, P15), testable (P1, P2, P5), concise (P8, P17), and unambiguous (P14, P15) attributes. Imperative language usage was considered as harmful for the following attributes: testable (P1, P18), small (P5), understandable (P7), and concise (P17). However, P12 and P16, who use scenarios with a technical approach in mind, considered declarative form of

writing harmful for understandable (P12, P16), estimable (P16), feasible (P16), testable (P16), unambiguous (P16), and valuable (P16) attributes. Imperative writing was considered a good practice that enhances th following attributes: estimable (P12, P16), understandable (P12, P16) feasible (P16), testable (P13, P16), unambiguous (P16), and valuable (P16).

The use of a third person point of view, exemplified in the step *When the administrator approves the task* in opposition to the first person step *When I approve the task*, was also pointed out by some participants. P18 highlights the use of third person as a good thing as *it could get your whole team to think like the user and that would be very useful.* The use of all forms of third person point of view, either referring to *the administrator* or *the user*, is considered as a good practice and enhances the understandable (P2, P7, P17), concise (P9), unambiguous (P6), and valuable (P14, P18) attributes.

Steps characteristics look at how many steps a scenario has, how long they are, and what step keyword to use. Having long scenarios with many steps is considered harmful mainly for concise (P2, P3, P6, P7, P8, P9, P15, P17, P18) and small (P1, P3, P4, P5, P10, P11, P13, P14), as already discussed. There were also reports about it affecting unambiguous (P3), understandable (P14), and testable (P18). In a similar way, lengthy statements are considered harmful for concise (P3, P5, P7, P9, P17), small (P1, P3, P4, P10), and understandable (P17). P17 says that *the more imperative you write your scenarios the more lines it will have, so if you have too many lines, you can kind of guess, you can probably state some of those things a little bit better*, which could indicate that having fewer steps is not a proper goal, but having scenarios written in a declarative format rather than imperative would.

Additionally, there was a concern with the natural step order being violated. P2 said that alternating the use of *When* and *Then* steps *means that you need to split up in two smaller scenarios* and that *a good test is exactly one verification and alternating When and Then [steps] means you are testing more than one thing in the same scenario.* Therefore, this violation of the natural step order is considered a bad practice that affects the following attributes: understandable (P2, P4, P8, P10, P14, P17, P18), concise (P2, P3, P8, P9, P14, P18), testable (P2, P7, P18), unambiguous (P3, P14, P17), valuable (P3, P18), and small (P6). In a similar way, the multiple repetition of the same step, demonstrated by the excessive use of steps in sequence or the "And" keyword, is also judged a bad practice. P4 summarizes it saying that *If there are many Ands then it is probably harder to understand.* This repetition of the same step bad-practice affects unambiguous (P4, P7, P11, P12, P14), concise (P4, P7, P8, P14), testable (P4, P6, P9), understandable (P4, P7, P14), and small (P5) attributes.

We also asked the participants to review the scenario title and feature file description when evaluating a given Diaspora's BDD scenario. Feature titles and descriptions were often pointed out as *intended business outcome or the intended business value* (P2) in one way or another, which would enhance the understandable (P4, P14) and valuable (P7) attributes. Regarding scenario titles, participants had declared it should either express the intended action (P3, P5, P9, P10, P13, P16, P17) or the intended outcome (P2, P8, P12, P14), or both (P18, P11). Only a few have declared that it affects an attribute – from those who did, they mapped it with understandable (P9, P18), testable (P2, P15), and concise (P12) attributes.

Despite the scenario's language, titles and steps descriptions, Gherkin language allows other types of constructions such as the use of a background section, tags, scenario outlines examples, data tables or double quotes aid to pass parameters. P15 reflects on the need of a background section, when asking himself: *does adding a Background make it harder for people to read? If it makes it harder for people to read, then 'no', it's a bad idea. If you've got a quite large number of scenarios, the Background should have a different title as well.* The use of background section can enhance the concise (P2, P4, P17, P18) and understandable (P2, P4, P17) attributes.

Tags were reported as useful for scenario's categorization (P1, P2, P3, P11, P12, P13, P14, P16) and as a communication tool (P3, P7, P10, P15, P17). The use of tags positively affects valuable (P1, P3, P4, P14, P17), understandable (P3, P11), prioritized (P3, P14), and unambiguous (P3) attributes. P2 says that tags can harm the understandable attribute due to their technical nature.

Data tables caused different reactions from participants. As summarized by P8, *tables has pros and cons. The pros are for repetitive tests that needs to cover a different range in the input and they could be considered one scenarios. The cons are that you might accidentally bunch many scenarios together that should be kept separate - so it should be less and less easy to read, less clear.* The use of data tables affect concise (P1, P3, P11, P17, P18), understandable (P3, P7, P17, P18), unambiguous (P14), and testable (P14) attributes.

Scenario outlines have also received mixed reactions from participants. P5 uses it in a pragmatic way, as stated: *So we used these scenarios outlines with examples to check the different type of cases. It's a really easy way to get all of this, to test all of these cases, to check if all the different options are covered and are correct. But I also understand how, you know, the fact that they are so easy to use and it's so easy to add more test cases is also a risk because then you end up with huge test suites and you have to be critical if each example actually add value.* The use of scenario outlines affects concise (P17, P18), understandable (P17, P18), unambiguous (P1), and testable (P17) attributes.

6 NEWLY-REDEFINED QUALITY CRITERIA AND PROPOSED QUESTION-BASED CHECKLIST

Our interview-based study revealed the participants' interpretation of the literature-informed attributes (previously presented in Section 5.1) as well as a set of personal criteria and their link to the former (reported in Section 5.2). We now consolidate these characteristics into a single list of newly redefined attributes to serve as input for our question-based checklist as summarized in Table 3. The final and consolidated list of newly-redefined attributes presented next in Section 6.1 answers RQ1. And, the proposed question-based checklist presented in Section 6.2 answers RQ2.

6.1 Newly-Redefined Quality Attributes

From the practitioners interpretations, their own personal criteria, and the reported relationships between them, we identified a final set of quality attributes to represent what a BDD scenario should be: essential, focused, singular, clear, complete, unique, ubiquitous,

Table 3: Newly-Redefined Attributes Mapping to Characteristics

Attribute	Characteristic
Essential	(Concise) Not Too Many Details
Essential	(Concise) Not Too Many Steps
Essential	(Concise) No Unnecessary Lines
Essential	(Small) Not Many Steps
Essential	(Steps) Few and Short Steps
Essential	(Steps) Steps Repetition
Essential	(Additional Constructions) Background
Essential	(Additional Constructions) Data Tables
Essential	(Additional Constructions) Scenario Outline
Focused	(Concise) Focused
Focused	(Testable) Focused
Focused	(Language) Declarative rather than Imperative
Focused	(Additional Characteristics) Declarative vs Imperative
Singular	(Small) Test One Single Thing
Singular	(Testable) Clear Outcomes and Verifications
Singular	(Testable) Clear and Simple Given's
Singular	(Unambiguous) Single Clear Intention
Singular	(Unambiguous) Scenarios Testing the Same Thing
Clear	(Language) Technical Jargon
Clear	(Concise) Clear
Clear	(Unambiguous) Vague Statements
Clear	(Unambiguous) High Granularity Steps Descriptions
Complete	(Testable) Follow the Steps
Complete	(Testable) Completeness
Complete	(Understandable) Self Contained
Complete	(Unambiguous) Completeness
Complete	(Additional Characteristics) Complete
Unique	(Valuable) Unique
Unique	(Valuable) Business Value
Unique	(Additional Characteristics) Cohesion
Unique	(Titles) Feature Description
Unique	(Titles) Scenario Titles
Unique	(Additional Constructions) Tags
Ubiquitous	(Understandable) Ubiquitous
Ubiquitous	(Language) Ubiquitous
Ubiquitous	(Language) Actor Consistency
Ubiquitous	(Unambiguous) Ubiquitous
Ubiquitous	(Additional Characteristics) Writing Patterns
Integrous	(Understandable) Good English
Integrous	(Additional Characteristics) Integrity
Integrous	(Steps) Steps Order

and integrous. These are new labels to characteristics already considered by the literature-informed attributes or by the participants themselves.

The *essential* attribute represents the fact that only essential information should be written into textual BDD scenarios. It is derived from interpretations of concise, which referred to avoid unnecessary details and steps, such as those found in Table 3: the "not many steps" characteristic, assigned to concise and small attributes; the steps' "unnecessary lines" and "many steps" characteristics; the background, data tables and scenario outlines additional constructions characteristics.

The *focused* attribute represents the need of declaring "what" a scenario should do (writing it in a declarative way), rather than describing "how" that action will be performed (writing it in an imperative way). Derived from the "declarative rather than imperative" characteristic in Table 3, that appears on the language and steps groups and as the focused interpretation of concise and testable literature attributes.

The *singular* attribute represents the need of a scenario to have a single purpose and to demonstrate this purpose clearly. It is derived from the following characteristics in Table 3: the "single clear intention"; the "test one thing"; the scenarios written differently were "testing the same thing"; clear outcomes (assigned to verifications on *Then* steps); and pre-conditions (assigned to *Given* steps).

The *clear* attribute appeared due to the fact that vague statements can harm as much as an excess of details. Additionally, high granularity steps could be easily identified as vague steps for technical people. Also, the use of technical jargon could represent less clear steps for business people – therefore, there should be a certain balance that allows a scenario to be correctly understood by all parties involved. This attribute is composed of the following characteristics from Table 3: the "technical jargon" language characteristic; the "clear" interpretation for concise attribute; the "vague statements" interpretation for unambiguous attribute; and the "high granularity" interpretation of unambiguous attribute.

The *complete* attribute can be seen from multiple perspectives. On the scenario level, all the information needed to understand and follow those steps should be present – represented by the "follow the steps" and the "self contained" characteristics in Table 3. On the feature level, the set of scenario's should provide enough coverage for that feature – an additional attribute in Table 3.

The *unique* attribute can be summarized by the quote *is it testing something fundamentally different to the other scenarios* from P15. This attribute represents how evident a scenario's business value is from its description and how interesting the scenario is – two characteristics in Table 3 that came from interpretations of the valuable attribute. Cohesion characteristic in Table 3 reinforces that idea, as it is interpreted as *the reason [for a scenario] to belong to a given feature file.* Additionally, each scenario title could inform the reader about its behavior, expressing its action and its outcome while correlating those with the actual steps. In a similar way, feature file descriptions can aid on recognizing a set of scenarios importance. Finally, tags, serving as the scenario's meta-data, can provide important information to express a scenario's value.

The *ubiquitous* attribute represents the consistent use of business terms, either taken from a glossary or a team's common knowledge, and helps to reinforce the need to bring scenarios closer to technical and business people alike. The characteristic of using business defined roles and/or personas in a consistent way enhances the ubiquity of a scenario description, according to some participants (P2, P6, P7, P17).

Table 4: Question-based Checklist for BDD Scenarios

ID	Question	Scope	Attribute
1	Can the feature file business value or outcome be identified by its description?	Feature	Unique
2	Does the feature file has any missing scenarios?	Feature	Complete
3	Does the scenario carry all the information needed to understand it?	Scenario	Complete
4	Does the scenario has steps that can be removed without affecting its understanding?	Scenario	Essential
5	How different each scenario is from the others?	Scenario	Unique
6	Can the scenario single action be identified on its title and match what the scenario is doing?	Scenario	Singular
7	Can the scenario outcome or verifications be identified on its title and match what the scenario is doing?	Scenario	Singular
8	Does the scenario respect Gherkin keywords meaning and its natural order?	Scenario	Integrous
9	Does the step correctly employs business terms, including a proper actor?	Step	Ubiquitous
10	Does the step has details that can be removed without affecting its meaning?	Step	Essential
11	Does the step express "what" it is doing by being written in a declarative way?	Step	Focused
12	Does the step allow different interpretations by being vague or misleading?	Step	Clear

The *integrous* attribute remind us that a scenario should respect the rules of Gherkin language, as the natural sequence of steps (*Given* steps first, followed by *When* and *Then* steps). Also, it should use proper steps tenses (*Given* steps in the past, *When* steps in the present, *Then* steps in the future), and it should respect each keyword type (*Given* describing pre-conditions, *When* describing actions and *Then* describing verifications) as described on both the understandable and the missing integrity attribute.

6.2 Question-based Checklist

Taking those newly labeled attributes into consideration, the question-based checklist in Table 4 was organized to guide practitioners on their evaluation of BDD scenarios. To use the checklist, we recommend the evaluator to answer Questions 1 and 2 by looking at the bigger picture first – the *feature file* as a whole, then proceed to analyze *every scenario* with Questions 3 to 8, and *every step* with Questions 9 to 12. These different scope levels emerged naturally from the study. Also, the output of those questions are meant to help the evaluator decide how good her BDD scenarios are and, if not satisfactory, may be the input of additional conversations around those scenarios.

To aid the presentation of the Question-Based Checklist, we will use the Feature file in Figure 2 as our evaluation target, introducing at the same time a simple proof of concept to demonstrate the checklist's feasibility of use.

Question 1's evaluation focuses on the feature file description, so the evaluator can check if the intended business outcome or business value is expressed there – a characteristic of the unique attribute. The feature file in Figure 2 represents the action, to block a user, but not what the benefit will bring to the business. By reading the scenarios' steps *Then I should not see any posts in my stream*, we can understand what the outcome of the block a user action is, but this is not represented in the feature file description. One could come back to the scenario writer with that suggestion in mind.

The purpose of Question 2 is to motivate the evaluator to check "what the scenarios are testing", in the hope that the evaluator can identify missing ones that should be part of that feature file coverage. It is apparent from Figure 2 that no negative scenarios are listed (when an user hit one of those block buttons but cancel the action). Also, no complex scenarios are listed – Alice is a friend with Bob, who decides to block her and do not see any of her posts anymore, but what happens with the posts of other Bob's friends? Those additional scenarios could be added to the same feature file.

Question 3 indicates to the evaluator if scenarios need any additional information to be understood – an evaluator should be able to "follow the steps". However, scenarios are not meant to be as complete as test cases descriptions, so a certain balance has to be achieved. All scenario's listed in Figure 2 are easy enough to follow – an interaction with the block button needs a confirmation, and once that happens and an user come back to the home pages, no posts are seen.

Question 4 targets unnecessary steps, that adds more information than what is essential to validate the behavior being tested. Assuming that "my stream" in Figure 2 is only seen in the user's home page, all the steps *When I go to the home page* could be safely omitted. If that assumptions is not true, calling "my stream" as "my homepage's stream" would also eliminate the need of the beforementioned steps. Additionally, the "profile block button" could be phrased as "Alice's profile block button", to remove the first *When* step on the second scenario in Figure 2. Finally, the evaluator should notice how well the background section was used to avoid repeating *Given* steps in the two scenarios.

Similarly, Question 5 aims to question the need of a scenario in that feature file. How unique a scenario is may be also be answered by the completeness analysis in Question 2, but Question 5 focuses on how different a scenario's business value is from the others scenarios and how important it is to the whole feature file. As the "block action" in Figure 2 can be performed from two locations, it makes sense to have two scenarios representing both of those variations.

The evaluator is asked, in Question 6, to focus on how the scenario single action, represented in a single *When* step, is aligned with the scenario title. In Figure 2's first scenario, the title lacks the information about where the user is when pressing the block button, an information that in the second scenario title's is answered by the "from the profile page" part. Also, in the second scenario, there are multiple *When* steps, already revealed as unnecessary by Question 4.

In Question 7 the evaluator is asked to focus on the scenario verification and outcomes, found on the *Then* steps, and in how aligned they are with the purpose expressed in the title. That outcome is missing in both scenario's titles in Figure 2 and need to be added there.

Question 8 validates the integrity of the scenario's keywords on every scenario, allowing the evaluator to question if *Given* steps are really representing pre-conditions, if *When* steps are representing actions and if *Then* steps are representing outcomes. If necessary, the evaluator can question the statements tenses as well. All steps in Figure 2 seem to respect the scenario's keywords, but all of them are written in the present tense and could be further enhanced by correcting it.

Question 9 allows the evaluator to ponder on the use of correct business terms in a consistently way in the steps, which would empower scenarios with an ubiquitous language that technical and non-technical people could understand. Additionally, the actor performing every step action should represent a business role rather than "I" or "the user", which is not the case in Figure 2, where all the subjects of the statements are "I" and could be changed to "Bob", the signed user according to the background section.

Question 10, similar to Question 4, aims to reduce unnecessary information, but on the step level rather than on the scenario level. If some details are important to be added to represent how different the scenario data is from others, maybe a data table would be needed. If a data table is present, the evaluator should question what is the need of it to understand the step description. In Figure 2 scenarios, the users' e-mails are irrelevant to the action being represented, as well as the "#unblock_user_button" technical name. Also, the data table in the background section is unnecessary, as the step on line 8 already describes that Alice and Bob are connected.

Question 11 brings the evaluator to question how focused a step is on the "what" is being done rather than in expressing "how" it is doing that, checking if the step is written in a declarative way. With the removal of the unnecessary steps suggested in Question 4 and the unnecessary details pointed out in Question 10, with Question 11 one can question if the "block" action need to be represented by the click of a button and the confirmation of an alert, two indications that the scenarios are tied to a web application.

Finally, Question 12 aims to balance the need to express actions with fewer words and details with the need to make the scenario understandable for every team member involved. If a step is too abstract or vague that makes the evaluator confused about what would be the actual user flow to perform that step, than it needs to be clearer than it is. As the scenarios in Figure 2 already declares where the user is when the block button is pressed, either in her home page or on the other user's home page, the user flow can be precisely understood, leaving no room to vagueness.

7 CONCLUSION AND FUTURE WORK

Our main goal was to propose a question-based checklist to evaluate BDD scenarios based on newly labeled quality attributes constructed from the knowledge of practitioners, in the hope it enhances practitioners' ability to evaluate their scenarios' quality by providing them with an standard guideline that can be the input of conversations to refine those scenarios. This goal was accomplished by arranging characteristics, taken from an interview-based study with 18 practitioners, into newly labeled quality attributes in Section 6.1 and formatting them into a question-based checklist in Section 6.2, answering our RQ1 and RQ2, respectively. We believe those outputs can aid future practitioners evaluations of their own BDD scenarios and help them to avoid known requirement problems, such as incomplete, underspecified and inconsistent requirements [4].

We still need to come back to practitioners and validate in practice how useful our quality attributes and question-based checklist can be. We deliberately avoid validating the question-based checklist before its relationship with the attributes and its structure have become stable enough to be used in real-world BDD contexts and its results could be trustworthy. This will be our next research step.

ACKNOWLEDGEMENT

We would like to kindly thank the study participants. Also, to acknowledge that the results presented in this paper were achieved in cooperation with Hewlett Packard Brasil LTDA, using incentives of the Brazilian Informatics Law (Law nº 8.2.48 of 1991).

REFERENCES

[1] Alistair Cockburn. 2000. *Writing Effective Use Cases.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
[2] Mike Cohn. 2004. *User Stories Applied: For Agile Software Development.* Addison-Wesley Publishing Co., Inc., Redwood City, USA.
[3] Daniela S. Cruzes and Tore Dyba. 2011. Recommended Steps for Thematic Synthesis in Software Engineering. In *Int'l Symposium on Empirical Software Engineering and Measurement, Banff, Canada.* 275–284.
[4] D. Méndez Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, T. Männistö, M. Nayabi, M. Oivo, B. Penzenstadler, D. Pfahl, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, R. Spinola, A. Tuzcu, J. L. de la Vara, and R. Wieringa. 2017. Naming the pain in requirements engineering. *Empirical Software Engineering* 22, 5 (2017), 2298–2338.
[5] Diaspora Foundation. 2017. Diaspora: The online social world where you are in control. https://diasporafoundation.org/ Visited in: 2018-02-03.
[6] M. Gartner. 2012. *ATDD by Example: A Practical Guide to Acceptance Test-Driven Development.* Addison-Wesley Professional.
[7] Petra Heck and Andy Zaidman. 2017. A framework for quality assessment of just-in-time requirements: the case of open source feature requests. *Requirements Engineering* 22, 4 (2017), 453–473.
[8] Ville Heikkila, Daniela Damian, Casper Lassenius, and Maria Paasivaara. 2015. A Mapping Study on Requirements Engineering in Agile Software Development. In *Euromicro Conference in SEng, Funchal, Portugal.*
[9] IIBA. 2009. *A Guide to the Business Analysis Body of Knowledge (BABOK Guide) 2nd Edition.* International Institute of Business Analysis.
[10] IIBA. 2015. *A Guide to the Business Analysis Body of Knowledge (BABOK Guide) 3rd Edition.* International Institute of Business Analysis.
[11] G. Lucassen, F. Dalpiaz, J.M.E.M. VanDerWerf, and S. Brinkkemper. 2015. Forging high-quality User Stories: Towards a discipline for Agile Requirements. In *Int'l Requirements Eng. Conf., Ottawa, Canada.* 126–135. https://doi.org/10.1109/RE.2015.7320415
[12] Gabriel Oliveira and Sabrina Marczak. 2017. On the Empirical Evaluation of BDD Scenarios Quality: Preliminary Findings of an Empirical Study. In *Workshop on Empirical Requirements Engineering in conjunction with the International Requirements Engineering Conference.* IEEE, Lisbon, Portugal.
[13] John Smart. 2014. *BDD in Action: Behavior-Driven Development for the Whole Software Lifecycle.* Manning Publications, Shelter Island, NY. 384 pages.
[14] John Ferguson Smart. 2017. Broad brushes and narrow brushes: there's more to BDD than Given/When/Then. https://johnfergusonsmart.com/theres-bdd-givenwhenthen/ Visited in: 2017-12-06.
[15] Anselm Strauss and Juliet Corbin. 1990. *Basics of qualitative research: grounded theory procedures and techniques.* Sage Publications. 272 pages.
[16] Matt Wynne and Aslak Hellesoy. 2012. *The Cucumber Book: Behaviour-Driven Development for Testers and Developers.* Pragmatic Bookshelf. 336 pages.
[17] Yang-Ming Zhu. 2016. *Software Reading Techniques: Twenty Techniques for More Effective Software Review and Inspection.* Apress.