Checklist-Aided Refinement of BDD Scenarios by Early-Career Developers: An Experience Report in a Corporate Training Program

Natalya Marjana Goelzer
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, Brazil
natalya.goelzer@edu.pucrs.br

Sabrina Marczak
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, Brazil
sabrina.marczak@pucrs.br

Abstract

Behavior-Driven Development (BDD) has become a common practice in agile software development and is increasingly adopted in industrial training programs preparing early-career developers for professional roles. In such settings, novices are expected to understand and refine scenario specifications written by a Product Owner before implementation — a task that often exposes their difficulty in interpreting requirements and identifying ambiguities, missing information, or weak alignment between scenario titles and intended behavior. This paper reports an experience of using a question-based checklist as a pedagogical aid to support early-career developers in understanding software requirements during the refinement of BDD scenarios. The intervention was integrated into four development sprints in a corporate training program at a multinational IT company. We reflect on the experience by triangulating two complementary lenses on the same population: (i) participants' self-reported perceptions, previously analyzed by the authors, and (ii) the refinement comments they produced during ad-hoc and checklist-aided sessions. The two lenses converge: the deeper engagement with requirements that participants reported perceiving manifests in their refinement artifacts as attribute-anchored comments at three granularities, surfacing requirement issues that ad-hoc refinement left invisible. We share lessons learned for educators and corporate training programs designing onboarding paths for requirements-quality skills in agile environments.

Keywords

Verification and Validation, Quality of Requirements, Checklist, Experience Report, Novice Developers

1 Introduction

Industrial training programs are an increasingly important bridge between academic preparation and professional practice in software engineering. In such programs, early-career developers, often interns or junior professionals with limited prior experience, are expected to engage with real artifacts, real stakeholders, and real development cycles. Among the skills these developers must rapidly acquire is the ability to read, interpret, and refine requirements specifications collaboratively with business representatives, an activity that demands more than syntactic familiarity with the artifacts involved. In agile environments, Behavior-Driven Development (BDD) has been adopted as a shared language for describing system behavior and aligning communication between technical and business roles [10, 25]. BDD promotes collaboration among developers, testers, and stakeholders by using structured scenarios written in natural language to describe expected system behavior, typically expressed in the Given-When-Then format defined by the Gherkin syntax [1, 25]. Well-written scenarios help clarify requirements, reduce ambiguities, and support shared understanding across the team [25]. To realize these benefits, however, scenarios must adhere to quality attributes such as clarity, completeness, and consistency [22]. Oliveira et al. [22] proposed a specific set of eight attributes (essential, focused, singular, clear, complete, unique, ubiquitous, and integrous) to guide the evaluation of scenario quality, organized as a checklist of guiding questions structured at three levels of analysis: the feature file, the scenario, and the steps. In many industrial settings, including the one reported in this paper, the Product Owner (PO) writes the scenarios and the development team is expected to refine them collaboratively before implementation. This refinement requires a skill that early-career developers rarely bring with them: the ability to interpret a specification critically and to identify ambiguities, missing information, weak alignment between titles and intended behavior, or vague step descriptions [22, 23, 26]. Unlike syntax, this skill cannot be taught only through examples; it depends on conceptual support that helps novices notice what is unclear, name what is missing, and discuss what is ambiguous before development starts. Without such support, refinement tends to surface only the most obvious issues, leaving subtler problems to emerge later during implementation, where they generate rework, clarification meetings, and misunderstandings with the PO.

A checklist of guiding questions offers candidate support for this learning challenge. By externalizing the criteria that experienced reviewers carry implicitly, a checklist may help novices focus their attention on relevant quality concerns during collaborative refinement [11]. In a previous study [13], we investigated how early-career developers perceived the use of the checklist proposed by Oliveira et al.[22] during the refinement of BDD scenarios in a corporate training program. Participants reported that the checklist helped them pay attention to detail, identify missing information, and engage in more cohesive group discussions. Perception data alone, however, cannot tell whether the checklist also changed how developers actually engage with requirements during refinement.

This paper reports the pedagogical experience behind that intervention and reflects on it through a second, complementary lens: the refinement comments and mentor observations produced during the sprints. The experience was conducted across a corporate training program at a multinational Information Technology (IT) company, with 20 interns in the edition. In the edition, early-career developers refined BDD scenarios written by an experienced

Product Owner over four sprints before using them as a basis for implementation. The intervention had two phases: the first two sprints were conducted without the checklist, and the last two were guided by it. We reflect on the experience by triangulating two complementary lenses on the same population: (i) participants' self-reported perceptions, previously reported in [13], and (ii) the refinement comments and mentor observations recorded across the sprints. By doing so, we examine whether the deeper engagement with requirements that participants reported perceiving also manifested in the artifacts they actually produced.

The results suggest that the checklist was associated with changes in how participants approached BDD scenario refinement. Without the checklist, participants identified relevant problems, such as missing information, unclear business rules, uncertain actors, and undefined expected results. However, the comments emerged unevenly across groups and were often expressed as general doubts. With the checklist, the comments became more explicitly connected to BDD scenario quality attributes and were organized across three levels of analysis: feature, scenario, and steps. The data suggest that this structure supported participants in noticing issues related to missing scenarios, weak alignment between titles and expected behavior, vague step wording, unclear triggers, and scenario boundaries. Some quality dimensions, such as the Focused attribute, appeared in participants' comments only after the checklist was introduced. This pattern suggests that structured guidance may have helped novices distinguish different types of scenario quality problems during refinement.

This experience report contributes to the discussion on educational approaches for early-career software developers in three ways. First, we describe how a checklist of guiding questions can be integrated into existing agile training rituals as a pedagogical aid for requirements understanding, so that other corporate training programs can adapt and replicate the approach. Second, we illustrate how a triangulated reflection on perception and practice can strengthen the interpretation of educational interventions in real training contexts where controlled comparisons are not feasible. Third, we share lessons learned about what worked, what did not work as expected, and what we would adjust in future iterations.

The remainder of this paper is organized as follows. Section 2 presents the theoretical foundations supporting the experience. Section 3 describes the context, the pedagogical design, and how we observed the experience. Section 4 presents observations across the two phases and the triangulation of perceptions and practice. Section 5 discusses lessons learned. Section 6 addresses limitations, and Section 7 concludes the paper.

2 Background

2.1 Question-based Checklist for BDD Scenarios

Behavior-Driven Development (BDD) is a collaborative software development approach that aims to improve communication between technical and business stakeholders through shared requirement descriptions written in natural language [25]. Instead of describing system behavior only through technical specifications, BDD encourages teams to discuss examples of expected behavior before implementation, fostering a shared understanding of requirements among developers, testers, and business representatives [1].

In practice, BDD scenarios are written using the Gherkin syntax and organized through the *Given-When-Then* structure, which represents the system context, the triggering action, and the expected outcome of an interaction [26]. Although this structure may support requirement clarification [21, 25], writing high quality scenarios remains challenging for early-career developers, who must interpret business goals, identify missing information, avoid ambiguities, and maintain consistency among related scenarios. As a result, novice teams may produce scenarios with vague descriptions, duplicated information, unclear business rules, or inadequate levels of detail, which can generate misunderstandings during implementation and increase the need for clarification meetings with PO.

To support the inspection of BDD artifacts, Oliveira [22] proposed a question-based checklist composed of 12 guiding questions for assessing the quality of BDD scenarios, based on quality attributes identified in professional practice. According to the author, well-written BDD scenarios should avoid unnecessary details (**Essential**), describe what is expected rather than how it should be implemented (**Focused**), and formalize a single intention (**Singular**). They should also be understandable to all parties involved (**Clear**), contain the information needed to cover the feature adequately (**Complete**), differ from one another to avoid redundancy (**Unique**), and use business terms and roles consistently (**Ubiquitous**). In addition, scenarios should respect the rules of the Gherkin language (**Integrous**). Together, these attributes provide an vocabulary for examining scenario writing and help teams externalize quality criteria that might otherwise remain implicit in individual experience.

The checklist is organized as a reading technique to guide collaborative inspection activities across three levels of analysis: the feature file, the individual scenario, and the steps. Questions 1 and 2 are intended for the feature file level, encouraging reviewers to first examine the broader structure of the feature as a whole. Questions 3 to 8 focus on the scenario level and should be applied to each scenario individually. Finally, Questions 9 to 12 address the steps level and should be used to inspect each step in detail. This progression allows reviewers to move from a broader view of the set of scenarios to a more detailed analysis of each scenario and its steps, making the inspection process more systematic and easier to apply in collaborative refinement activities.

2.2 Checklists as Pedagogical Supports

Checklist guided inspection techniques have long been used in Software Engineering as mechanisms for defect detection and quality assurance in software artifacts [9, 17]. Traditional inspection approaches rely on structured reading techniques and predefined verification criteria to support systematic artifact evaluation [24]. In contrast to ad hoc reviews, inspections guided by checklists may reduce dependence on individual experience by directing reviewers toward specific quality concerns [17] and improving the consistency with which ambiguities, inconsistencies, and missing information are identified across different artifact types [12]. Previous research has reported this kind of inspection in several Software Engineering domains, including business process modeling [7], UML diagrams [2], software product lines [8], IoT requirements [16], BDD refactoring [15], test plan evaluation [3], Scrum assessment [6], usability

evaluation [4], and privacy compliance [19]. Together, these studies suggest that the approach can be adapted to different artifact types and development activities while supporting more systematic inspection practices.

Beyond their role in quality assurance, checklists have also been discussed as instruments that support professional learning and decision making. Gawande [11] argues that checklists may function not only as verification tools but also as cognitive supports that help practitioners manage complex activities through structured guidance, particularly when the underlying judgment depends on criteria that practitioners have not yet internalized through experience. This perspective is relevant in educational and onboarding contexts because novice professionals are still developing the mental schemas required to evaluate artifacts systematically. In BDD refinement activities, interns and junior developers may struggle to recognize problems related to ambiguity, completeness, and scenario scope, since these evaluations depend on familiarity with both technical and business concepts. In such contexts, a checklist composed of guiding questions may act as an external support that organizes collaborative discussions and externalizes the quality criteria experienced reviewers apply implicitly, helping novices focus their attention on relevant quality concerns during refinement activities. Despite this potential, the use of checklists as instructional supports for early career developers in BDD training contexts remains comparatively underexplored in the Software Engineering education literature.

2.3 Related Work

Previous studies have investigated BDD adoption in different contexts. Pereira [23] analyzed practitioners' perceptions of the benefits and limitations of BDD in industry, and Nascimento [18] discussed positive and negative impacts of its adoption in agile teams, including difficulties related to scenario interpretation and maintenance. Closer to our setting, Couto et al. [5] reported benefits perceived by novice teams—particularly regarding the use of ubiquitous language for requirement alignment—but also observed that beginners still struggled to interpret business rules and structure scenarios consistently, which suggests that learning BDD is not only a matter of practice but also of having adequate support to evaluate scenario quality. This concern is reinforced by Oliveira et al. [21], who argued that poorly written scenarios compromise communication and implementation, and by Oliveira and Marczak [22], who identified quality attributes for BDD scenarios and proposed a question-based checklist to support their systematic evaluation [20]; the authors suggested the checklist could be especially useful for beginners but acknowledged that its effectiveness had not been empirically assessed in real training environments. Our previous study [13] took a first step in this direction by examining participants' perceptions of the checklist in a corporate IT training program conducted in partnership between a university and a company. The present experience report extends that investigation by analyzing how the checklist actually manifested in practice: we describe how the checklist was integrated into agile training rituals across four sprints, triangulate participants' perceptions with the refinement artifacts and review comments they produced, and discuss lessons learned for corporate onboarding programs in BDD and requirements quality.

3 The Checklist-Aided Refinement Experience

3.1 Case Context

The experience reported in this paper took place in a Training Program conducted through a long-term partnership between University and a multinational IT company, here referred to as ORG due to contractual restrictions. The program prepares undergraduate students from areas related to computing for professional software development by combining technical training at the university with practical experience in an industry context [14]. It is organized into three phases: technical training at the university, a *hands-on* phase in which participants develop a real internal software system proposed by ORG, and integration into ORG development teams. The experience reported here was conducted during the *hands-on* phase, when participants move from guided training to collaborative development under conditions closer to professional practice.

The class involved 20 interns who voluntarily participated in the experience. They were undergraduate students from different higher education institutions, mostly in the early stages of their academic formation. Most participants classified themselves as beginners in software development, with up to three years of experience. Previous contact with BDD and requirements practices was limited: 85% reported no previous knowledge of BDD, none had prior experience writing or executing BDD scenarios, and they had not participated in formal scenario inspections before the intervention.

During the *hands-on* phase, the interns worked as the development team of a real project. The team also included one Product Owner (PO), responsible for writing the system functionalities as BDD scenarios, and two mentors, who supported the participants in agile practices, self management, and technical or process matters. One of these mentors was also the author of this study and accompanied the activities throughout the experience. The project followed an agile cycle of four two week sprints, each beginning with a planning meeting in which the PO presented the BDD scenarios, and ending with a demonstration to the PO followed by a retrospective. For the inspection activities, the 20 interns were organized into four groups of five participants, with composition varying across sprints to preserve the pedagogical emphasis on autonomy, collaboration, and peer learning. The inspected scenarios were not created for research purposes: they were written by the PO as part of the regular sprint planning process and connected to the product backlog of the internal system under development.

3.2 The Design of Intervention

The design of the experience was based on the idea that early-career developers need explicit support to evaluate the quality of BDD scenarios before using them as a basis for implementation. The checklist was introduced as a learning support for collaborative refinement. Its purpose was to guide the interns' attention toward concrete aspects of scenario quality and to support group discussion about ambiguities, missing information, unnecessary details, or inconsistencies. This choice was aligned with the role of structured reading techniques in software inspection, in which specific

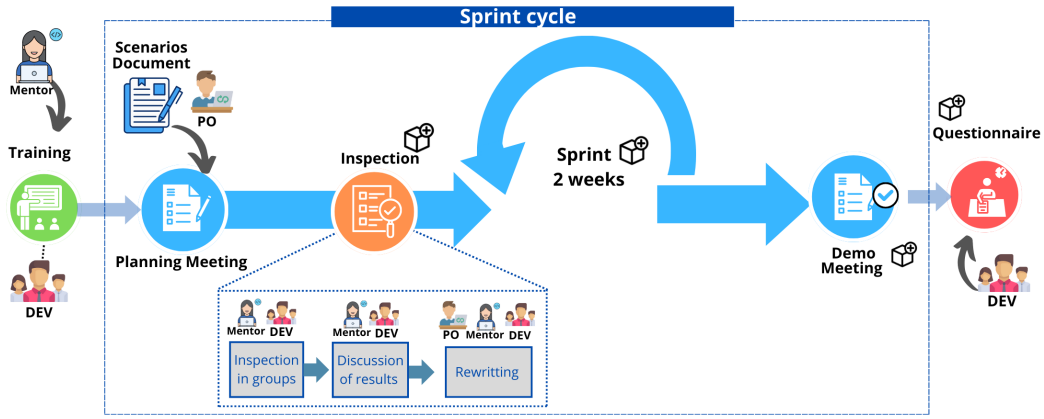

Figure 1: Cycle adopted for the BDD scenario refinement activities across the four sprints.

Table 1: Number of features and BDD scenarios inspected across sprints.

Phase	Sprint	Features	Scenarios
Without checklist	Sprint 1	5	9
Without checklist	Sprint 2	5	14
With checklist	Sprint 3	4	14
With checklist	Sprint 4	1	6

questions can guide reviewers in the identification of defects in artifacts [17].

The selected instrument was the checklist proposed by Oliveira [20] for evaluating the quality of BDD scenarios. This checklist was considered appropriate for the training context because it is directly aligned with the structure of BDD artifacts and provides concrete questions for scenario assessment. The intervention was conducted during the hands-on phase of the training program, across four development sprints. In total, 43 BDD scenarios were inspected. These scenarios were written by the PO as part of the regular sprint planning activities, based on the product backlog. No additional scenarios or artificial tasks were introduced for research purposes.

The experience was organized into two phases. In Sprints 1 and 2, the groups inspected the BDD scenarios without the checklist, relying on their own understanding and group discussion. In this phase, 23 scenarios were inspected. In Sprints 3 and 4, the same refinement activity was conducted with the support of the checklist, totaling 20 inspected scenarios. The 20 interns participated in the activities throughout the intervention and worked in groups of approximately five members. At the beginning of each sprint, group composition was partially reconfigured to broaden peer interaction, while preserving the program's pedagogical emphasis on autonomy and collaboration. The distribution of features and scenarios across the four sprints, summarized in Table 1, reflects the natural variation of the product backlog and was defined by the PO independently of the research design.

Each phase was preceded by a training session conducted by one of the mentors. The first training prepared participants to inspect scenarios without a structured instrument, emphasizing the identification of unclear, incomplete, or inconsistent parts of the scenarios. The second training introduced the checklist, explained its questions and quality attributes, and described how groups should record their observations in the standardized spreadsheet. The PO did not participate in these training sessions, since this professional already had more than four years of experience in the role and was responsible for preparing the scenario documents as part of the regular project workflow.

Figure 1 summarizes the cycle adopted in each sprint. After the sprint planning meeting, in which the PO presented the BDD scenarios, the interns inspected the scenarios collaboratively in their groups. During Sprints 1 and 2, comments were registered directly in the scenario documents. During Sprints 3 and 4, comments were registered in the checklist spreadsheet. All inspection sessions were supervised by the mentor, who supported the process and recorded field notes.

After the group inspection, the 20 interns and the mentor participated in a class-wide debriefing session to align interpretations, discuss uncertainties, and consolidate the observations. The groups and the mentor then presented the questions to the PO, who reviewed the comments and adjusted the functionality descriptions when necessary. The activity was integrated into the development cycle, giving it a dual purpose: it supported the interns in learning how to reason about scenario quality, and it contributed to the project workflow by creating a structured moment for clarification before development continued.

3.3 How We Observed the Intervention

To reflect on the experience, we combined two lenses. The first was based on participants' perceptions, collected through an online questionnaire applied at the end of each two sprint phase. The first questionnaire focused on the difficulties faced when inspecting scenarios in an ad hoc way, and the second focused on the clarity, applicability, perceived usefulness, and possible reuse of the checklist. These questionnaires were previously analyzed in [13] and are used here to explain how the participants interpreted the activity. The second lens was based on practice: the comments produced by the groups while reviewing the scenarios, recorded directly in the scenario documents in the first phase and in the checklist spreadsheet

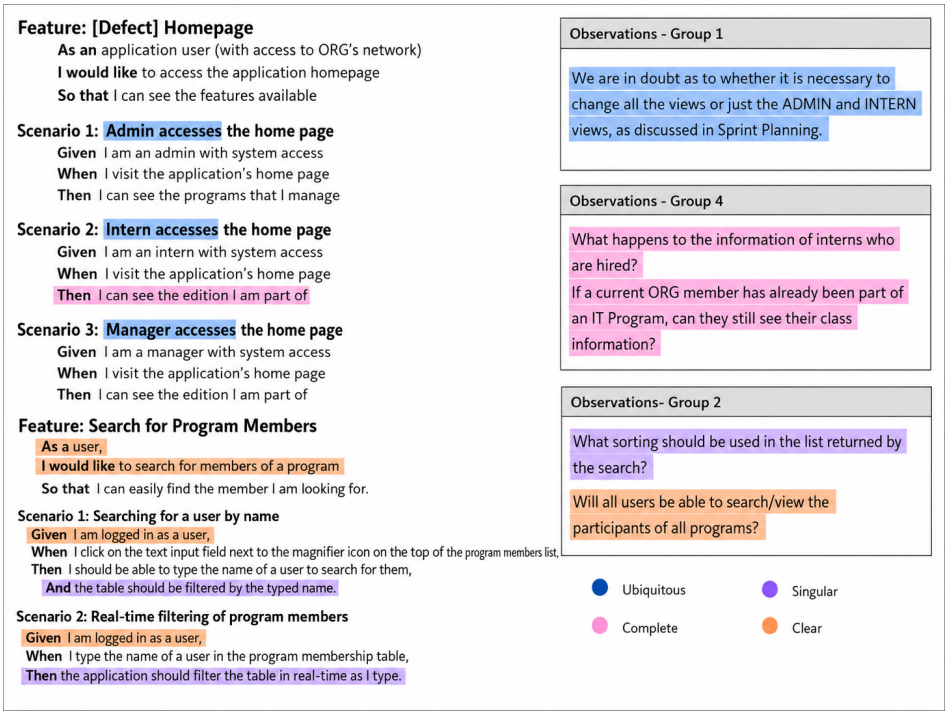

Figure 2: Illustrative example of BDD scenario evaluation by groups without checklist use.

in the second phase. The mentor's logs provided contextual support, with notes on group behavior, retrospective discussions, burndown information, and extra clarification meetings with the PO. To organize the reflection, we grouped the observations according to the eight quality attributes proposed by Oliveira [22]. In the phase without the checklist, the comments registered in the scenario documents were associated with the corresponding attributes after the sprint activities. In the phase with the checklist, the spreadsheet already linked each observation to one of the inspection questions, and therefore to the quality attributes. This allowed us to compare how participants reasoned about scenario quality in both phases. Due to confidentiality restrictions with ORG, the complete set of inspected scenarios cannot be disclosed. In the following section, we present selected and adapted examples that preserve the patterns observed during the experience while avoiding sensitive project information.

4 Observations from the Experience

The analysis strategy focused on categorizing the observations made by participants during the inspection of BDD scenarios, with reference to the quality attributes proposed by Oliveira et al. [22]. In Phase 1 (without checklist), comments were recorded directly in the scenario documents and later tagged by Author 1, with subsequent review by Author 2, who has greater expertise in requirements specification. In Phase 2 (with checklist), participant feedback was recorded in a structured spreadsheet, allowing each observation to be explicitly associated with one of the 12 checklist questions, which ensured greater traceability and consistency.

Although due to NDA restrictions with the ORG, the full set of inspected scenarios cannot be disclosed, selected examples were prepared for illustrative purposes. These examples aim to support the reader's understanding of the types of quality issues identified and how they were addressed during the inspection process. The scenarios chosen for illustration reflect real patterns observed during the intervention, but were adapted to ensure compliance with the non-disclosure constraints. The next subsection 4.3 presents a comparative analysis of the two phases, highlighting how the checklist influenced the inspection process and supported the identification of quality issues in BDD scenario writing.

4.1 Inspection without a checklist

Before the checklist was introduced, the refinement activity was conducted through group discussion and direct annotations in the scenario document. The interns reviewed the BDD scenarios based on their own understanding of the functionality, their interpretation of the sprint planning meeting, and the questions that emerged while reading the scenarios. This first moment allowed us to observe how refinement looked without structured guidance and which types of issues related to scenario quality early-career developers were able to identify through ad hoc discussion.

Figure 2 illustrates one example of this activity, showing the association between the areas highlighted in the scenario document, the groups' comments, and the quality attributes later used to organize our reflection. In this example, the comments were mainly associated with the attributes Ubiquitous, Complete, Singular, and Clear.

As an admin
I would like to view the results of a matchmaking process
So that I can be sure that the best distribution possible of all interns along the teams will be done

Scenario 1: Viewing the Distribution Report
Given that I am an admin and I am on the "Match Maker Report" page
When the algorithm has processed the distribution of the interns to the teams
Then I can see the result of the algorithm assigning each intern for each team in the best distribution possible

Scenario 2: Inconclusive Report
Given that I am an admin and the report is inconclusive
When the algorithm is unable to generate a conclusive distribution for all interns
Then I can see a partial report (can be more than 1) suggesting distributions to help me to solve the situation

Scope	Attribute	#	Question	Answer		Observations
				Yes	No	
Feature	*Unique*	1	The feature file business value or outcome can be identified by its description?		X	
	Complete	2	The feature file have any missings scenarios?		X	We believe that there's scenario missing
Scenario	*Complete*	3	The scenario carries all the information needed to understand it?		X	We don't know how to properly address this issue, but this scenario is confusing. It is the scenario that implements the algorithm by itself but it is also a user view inside the application? Maybe another User Story should be made to configure the algorithm and only the algorithm. Also, there should be a scenarios on when to run the algorithm (is it when the admin press a button? Or when the last intern submits its preferences? We don't know)
	Essential	4	The scenario has steps that can be removed without affecting its understanding?	X		
	Unique	5	How different each scenario is from the others?			They are similar because they share the same core functionality
	Singular	6	The scenario single action can be identified on its title and match what the scenario is doing?		X	As stated above, on question #3, this scenario is confusing
	Singular	7	The scenario outcome or verifications can be identified on its title and match what the scenario is doing?		X	As stated above, on question #3, this scenario is confusing
Steps	*Integerous*	8	This scenario respects Gherkin keywords meaning and its natural order?	X		
	Ubiquitous	9	Does that step correctly employ business terms, including a proper actor?	X		
	Essential	10	Does that step have details that can be removed without affecting its meaning?		X	
	Focused	11	Does that step express "what" it is doing by being written in a declarative way?		X	The steps does not contain a good definition for when to run the matchmaking algorithm - "best distribution possible" may be vague.
	Clear	12	Does the step allow different interpretations by being vague or misleading?		X	

Figure 3: Illustrative example of an inspection performed by a group with checklist support.

One type of observation concerned the alignment between the written scenario and what had been discussed during sprint planning. For example, one group commented: "*We are not sure as to whether it is necessary to change all the views or just the admin and intern views, as discussed in Sprint Planning.*" This comment was associated with the Ubiquitous attribute, since it pointed to uncertainty about the scope of the change and about which actors or views should be considered. The written scenario did not fully reflect the shared understanding built during the planning meeting, which created room for different interpretations during implementation.

Other comments indicated missing information about business rules and expected behavior. The question "*What happens to the information of interns who are hired? If a current ORG member has already been part of an Program, can they still see their class information?*" was associated with the Complete attribute, since the scenario did not provide enough information about how the system should handle user information in specific situations. Comments such as "*There are missing steps*" showed that, even without the checklist, participants were able to notice when scenarios did not provide enough information to guide implementation or validation.

Other groups raised questions related to the Singular and Clear attributes. The comment "*What sorting should be used in the list returned by the search?*" pointed to a lack of definition of the expected ordering, an issue tied to the Singular attribute since it could lead developers and stakeholders to assume different behaviors for the same functionality. Comments related to Clear, such as "*Will all users be able to search/view the participants of all programs?*" and "*Not clear which parameters can be edited*", showed that some scenarios did not make actions, permissions, or editable elements explicit enough. A smaller number of comments addressed the Unique attribute, especially when titles did not clearly represent the intended functionality, as in "*the title could be more detailed, including the word 'visualize', which represents the main action.*"

Across the two inspections conducted without the checklist, the comments varied considerably among groups. In the first inspection, Groups 1 and 2 identified one problem each, Group 3 did not identify issues, and Group 4 raised eight comments, two of which had already been mentioned by other groups, resulting in eight issues taken to the PO after discussion. In the second inspection, Group 1 did not identify issues, Group 2 identified one, Group 3 raised two, and Group 4 raised six questions, leading to six adjustments after discussion with the PO. Some issues were perceived by more than one group, suggesting that certain gaps were visible to different participants, while other issues were identified only by specific groups, suggesting that the activity depended heavily on each group's interpretation, attention to detail, and discussion dynamics. The absence of structured questions did not prevent the groups from identifying relevant problems, but the comments tended to emerge unevenly and were organized only after the activity, when they were associated with the quality attributes proposed by Oliveira et al. [22].

After each group inspection, the interns and the mentor aligned their understanding of the comments and prepared the questions

to be discussed with the PO, who clarified doubts and adjusted the wording when necessary. This sequence helped ensure shared understanding of the functionalities before implementation began, but also showed that, without structured guidance, refinement relied heavily on how each group interpreted the scenarios and on the later discussion needed to consolidate the comments.

4.2 Inspection with Oliveira's checklist

After the checklist was introduced, the refinement activity kept the same collaborative structure, but the way comments were produced and organized changed. Instead of registering doubts only as free annotations in the scenario document, the groups used a spreadsheet structured around the checklist questions. This made each observation closer to a specific quality attribute and to a specific level of analysis: the feature, the scenario, or the steps. The checklist did not remove the need for discussion, but it offered a more explicit path for examining the scenario writing.

Figure 3 illustrates one example of an inspection conducted with checklist support. In this example, the comments were associated with the attributes Complete, Singular, and Focused, showing how the participants used the checklist questions to connect their doubts to specific aspects of scenario quality.

At the feature level, the checklist helped participants question whether the set of scenarios was sufficient to represent the expected behavior. The comment "*We believe that there's a scenario missing*" was associated with the Complete attribute and indicated that participants were not only reading each scenario as an isolated text, but also considering whether the feature required additional scenarios or acceptance criteria.

At the scenario level, several comments showed uncertainty about the intention and boundaries of the scenario. One group registered the following observation: "*We don't know how to properly address this issue, but this scenario is confusing. Is it the scenario that implements the algorithm by itself, or is it also a user view inside the application? Also, there should be scenarios on when to run the algorithm (is it when the admin presses a button? Or when the last intern submits their preferences? We don't know).*" This comment was associated with the Complete attribute because it indicated missing information about the objective of the scenario, the separation of responsibilities, and the trigger for the expected behavior.

The Singular attribute appeared when participants questioned whether a scenario had one clear intention. Comments such as "*The scenario's unique action cannot be identified in its title, but it specified that the functionality will be applied on the preferences insertion page*" and "*The narrative outcome seems to be different from the outcome of the title*" suggest that, with the checklist, the groups paid attention not only to missing information but also to the alignment between title, narrative, action, and expected outcome. Related to this, the Unique attribute appeared in comments about titles that did not clearly represent the functionality or its result, as in "*It is not described in the title what can be done with a participant's profile*". Comments related to the Clear attribute pointed to vague wording or unclear references to the application context, as in "*The title could be more specific*" and "*In the title it doesn't say what to do with the 'where' field*". In these cases, the checklist supported a more direct discussion about how vague wording could affect interpretation.

A particular observation was that the Focused attribute appeared only after the checklist was introduced. The comment "*The steps do not contain a good definition for when to run the matchmaking algorithm (best distribution possible) may be vague*" indicated that the steps did not make clear when the algorithm should be executed and that the expression used to describe the expected result could be interpreted in different ways. This type of observation was not produced in the ad hoc phase, suggesting that some quality dimensions only became visible to early-career developers when the checklist explicitly prompted them. Similarly, the Ubiquitous attribute appeared in comments where participants identified missing or unclear actors, as in "*The title should also mention the managers who are the actors in the scenario*", pointing to the consistency of business terms and roles across the scenarios.

Across the inspections with checklist support, the activity remained collaborative. In the first inspection, the groups raised 13 questions in total. Convergence among groups was visible: Groups 1 and 3 reported common issues, and Groups 1, 3 and 4 raised concerns about the same scenario, suggesting that the checklist directed attention to similar quality concerns across teams. After group discussion, 8 questions were consolidated and taken to the PO. In the second inspection, Groups 1, 2 and 4 did not identify problems, and only Group 3 raised one issue, which was discussed and adjusted with the PO.

4.3 Triangulating Perceptions and Practice

In our previous work, we reported how participants perceived the use of the checklist after the refinement activities [13]. Their responses indicated that the checklist helped them pay attention to scenario quality and organize the inspection activity. Participants stated that "*the checklist helped identify specific areas for improvement*" (P36), that "*it makes it easier to identify missing or poorly described requirements*" (P5), and that "*the checklist helps to see the details of the scenarios in more detail*" (P4). Other participants described it as a guide for the activity, reporting that "*The checklist helped in understanding what to evaluate and how*" (P40) and that "*It was very useful to guide the evaluation, making the activity more organized and simple to carry out*" (P28). These perceptions suggested that the checklist supported attention to detail, reduced ambiguity, and helped participants understand what should be observed during scenario refinement.

When we looked at the comments produced by the groups during the refinement activities, we observed that these perceptions also appeared in the way participants discussed the scenarios. Without the checklist, comments were often written as general doubts about unclear business rules, missing steps, uncertain actors, or expected results. With the checklist, comments became more directly connected to the quality attributes proposed by Oliveira et al. [22], and this change was visible at three granularities: at the feature level, participants questioned missing scenarios and feature coverage; at the scenario level, they discussed the alignment between titles, narratives, unique actions, and expected outcomes; and at the step level, they pointed to vague wording, unclear triggers, and actions that could allow different interpretations. The perception

Table 2: Convergence between participants' perceptions and refinement practice.

Attribute	Perception lens	Practice lens	Convergence observed
Complete	Participants reported that the checklist helped them identify missing or poorly described requirements and missing specifications.	Groups pointed to missing scenarios, missing acceptance criteria, and insufficient information to understand when or how a functionality should be executed.	The perceived support for identifying gaps appeared in comments about feature coverage, absent scenarios, and incomplete business rules.
Clear	Participants reported that the checklist supported ambiguity reduction and made the subject clearer.	Groups commented on unclear titles, vague outcomes, unclear editable parameters, and lack of specificity about where actions should occur in the application.	The perception of reduced ambiguity was reflected in comments focused on wording, specificity, and alignment between title, narrative, and expected behavior.
Singular	Participants reported that the checklist helped them understand what to evaluate and how to conduct the inspection.	Groups questioned whether a scenario had one identifiable action and whether the title matched what the scenario was doing.	The checklist helped transform a broad impression of confusion into comments about the scenario's single action and intention.
Unique	Participants reported that the checklist helped them pay attention to details in the scenarios.	Groups observed that some titles did not represent the functionality or did not allow the expected result to be identified.	The perceived attention to detail appeared in comments about titles as elements that support traceability and scenario identification.
Ubiquitous	Participants reported that the checklist helped improve communication and shared understanding among team members.	Groups identified unclear or missing actors, inconsistent references to roles, and lack of clarity about which users were involved in the scenario.	The perceived improvement in communication was reflected in comments about actors, roles, and business terms used in the scenarios.
Focused	Participants reported that the checklist helped them evaluate scenario elements in a more organized way.	Groups identified steps that did not clearly express when an action should occur or that used expressions that could be interpreted in different ways.	The organization provided by the checklist supported attention to the step level, especially when actions and outcomes were not expressed in a declarative way.

that the checklist helped participants "*see the details of the scenarios in more detail*" (P4) was reflected in practice through more focused, attribute connected comments.

Table 2 summarizes this convergence between the perception lens and the practice lens. The table does not aim to quantify the effect of the checklist. Instead, it organizes how participants' reported perceptions were reflected in the refinement comments produced during the activity.

This triangulation helped us interpret the checklist as a pedagogical aid for making criteria of scenario quality visible during refinement. The participants' perceptions showed that the checklist was experienced as a guide for understanding, organizing, and discussing the scenarios, and the refinement comments showed how this guidance appeared in practice through observations connected to specific attributes and to different levels of the BDD artifact. At the same time, the comments also indicate that the checklist did not remove all uncertainty from the process. Some groups still identified few issues, and some doubts still needed to be discussed with the PO. The main change we observed was therefore not the elimination of ambiguity, but the creation of a more explicit structure for noticing, naming, and discussing ambiguity before implementation.

4.4 Contextual Evidence from the Sprints

To complement the inspection data, we also analyzed observational records kept by the program mentor across the four sprints, including burndown charts, logs of unscheduled clarification meetings with the PO, and retrospective notes. These records are not treated as isolated evidence of checklist effectiveness; rather, they offer contextual indications of how scenario refinement manifested in the teams' daily work.

Burndown Charts. All four sprints met their goals within the planned timeframe and without rework. The burndown charts show an earlier and more consistent ramp-up in the checklist phase. On the first day of work, Sprints 1 and 2 (ad hoc) recorded 8 and 0 hours respectively, while Sprints 3 and 4 (checklist) recorded 34 and 15.3 hours. The same direction held over the first three days, with the ad hoc sprints totaling 66 and 40 hours and the checklist sprints totaling 74.5 and 60.3 hours. We report this as an indicative observation, not a controlled measurement: the sample is small (two sprints per phase), Sprint 4 had a reduced scope (one feature, six scenarios), and participants gained product familiarity over time. The pattern is, however, coherent with what participants reported - *'having a better understanding of what needs to be done makes the process of starting the project much faster'* (P26) — suggesting that more understandable scenarios may be associated with a smoother start of execution.

Extra Clarification Meetings. Across the four sprints, 18 unscheduled meetings were held with the PO. In Sprints 1 and 2, most of them (8 of 12) addressed unclear or incomplete specifications, typically involving layout descriptions, missing acceptance criteria, or ambiguous business rules — issues that emerged during development from artifacts delivered in planning. In Sprints 3 and 4, the number of clarification meetings decreased to 6, and their nature shifted toward broader planning and prioritization questions, with doubts concentrated in the early days of each sprint and less linked to interpretation problems. The checklist did not eliminate the need

for clarification, but the observed shift suggests a possible reduction of ambiguity in the documentation and an earlier surfacing of open points, in line with the improvements in scenario precision and decreased late-stage clarification with the PO reported by [13].

Retrospective Notes. In the first two sprints, retrospectives surfaced difficulties tied to vague scenario descriptions, effort estimation, and unclear requirements, reinforcing the motivation for the intervention. In Sprint 3, the focus of the notes moved toward teamwork aspects (decision making and proactive task management) with fewer references to documentation issues. Sprint 4, with a reduced scope and overlapping with the onboarding for the next program phase, produced limited feedback. The relative absence of scenario related concerns in the retrospectives of the checklist phases is not conclusive on its own, but it is compatible with the perceptions reported in [13], where participants indicated that the checklist helped them understand functionalities, clarify sprint work, and reduce ambiguities in the documentation.

Taken together, these sprint level observations add a contextual layer to the two lenses discussed previously. They do not support a direct causal claim: the sprints differed in scope, number of scenarios, and project moment, and the teams gained experience with the product and with the refinement activity over time. Even so, the pattern suggests that refinement guided by the checklist was associated with a more structured discussion of scenario quality before implementation, an indication that is coherent both with participants' perceptions and with the comments produced during the inspection activities, and that may motivate further investigation in future studies.

5 Lessons Learned

This section summarizes five lessons drawn from combining participants' perceptions, refinement comments, mentor observations, and sprint records. For each, we note what we would adjust in a future iteration.

Lesson 1 — The checklist supports requirements understanding, not only scenario writing. Participants reported that the checklist helped them identify missing or poorly described requirements and pay attention to detail [13], and this perception was reflected in the refinement comments, which addressed missing scenarios, unclear titles, vague steps, and incomplete acceptance criteria. By externalizing the criteria experienced reviewers apply implicitly, the checklist gave novices a concrete path to slow down, examine the requirement, and name what they did not understand before implementation. The lesson is not that the checklist solves requirements understanding by itself (participants with prior familiarity reported less benefit and some doubts still emerged during development) but that it creates a shared structure for novices to ask better questions. In a future iteration, we would add more guided examples before the first checklist-supported inspection, showing how the same comment may map to different attributes (Complete, Clear, Singular, Focused) depending on the problem.

Lesson 2 — Some quality dimensions only become visible with structured guidance. Without the checklist, participants identified relevant problems unevenly across groups, mostly related to missing information, unclear business rules, actors, and expected results. With the checklist, comments became more specific and were distributed across three granularities (feature, scenario, step). The clearest case is the Focused attribute, which only appeared after the checklist was introduced: novices may sense that a scenario is confusing, but they do not spontaneously distinguish whether the problem is lack of completeness, lack of focus, weak title alignment, or vague step wording. The pedagogical implication is that attributes like Complete and Clear can be taught through examples, while attributes like Focused, Singular, and Unique require explicit prompts and supervised practice.

Lesson 3 — The mentor is part of the support system. The checklist did not work as a standalone artifact: the activity included training, group inspection, debriefing, alignment of interpretations, and discussion with the PO, and the mentor mediated each of these moments. The mentor's role was most useful when facilitating discussion — helping participants articulate what was unclear, comparing interpretations across groups, and transitioning inspection comments into PO conversations — rather than solving the ambiguity directly, which preserved the learning purpose of the activity. For replication, educators should plan the facilitator role explicitly, including time for debrief, a defined process for consolidating comments, and a moment to discuss them with the PO; without this mediation, the checklist risks becoming a form to fill in rather than a tool for shared understanding.

Lesson 4 — Integration into existing rituals matters more than the artifact itself. The checklist worked because it was placed between sprint planning and implementation, using the same scenarios that would guide development, rather than introduced as an isolated classroom exercise. Sprint records support this interpretation: clarification meetings with the PO became fewer and less focused on interpretation problems, and retrospectives contained fewer concerns about scenario documentation, observations that must be read with caution given that sprints differed in scope and participants also gained experience over time. For training programs, the recommendation is to embed the checklist in an existing ritual where requirements are already discussed, and to add an intermediate checkpoint during the sprint to capture doubts that only surface during implementation.

Lesson 5 — The checklist evolves through use. The pilot already led to adjustments (added practical training, refined inspection document, language alignment), and the main experience surfaced further participant suggestions: a field to evaluate the written narrative, simpler and less redundant questions, and better time budgeting [13]. The original instrument provided the conceptual structure, but its pedagogical use required adapting how it was introduced and operationalized and the partner organization continued to use the refined checklist in two subsequent cohorts, with approximately 40 additional developers. The actionable point for educators is to treat the checklist as an evolving pedagogical resource: review it after each cohort and adjust based on participant feedback, mentor observations, and stakeholder needs. We would also instrument effort, clarification, and rework data from Sprint 1 using a predefined template, and plan a longer observation window to examine whether the practice stabilizes over time.

6 Limitations of the Experience

This experience was conducted in a real training environment, which strengthened its practical relevance but also introduced limitations that should be considered when interpreting the observations reported in this paper.

A first limitation concerns the natural learning that occurred across the four sprints. The refinement activities without the checklist took place in the first two sprints, while the activities supported by the checklist took place in the last two sprints. During this period, participants became more familiar with BDD, with the product under development, with the PO's way of writing scenarios, and with the refinement activity itself. The scenarios inspected also varied across sprints, since they came from the regular product backlog and not from a controlled experimental design, with their number, scope, and complexity following the priorities of the project. For these reasons, the differences observed between the two phases cannot be attributed to the checklist alone, and the experience does not support a direct comparison of defect detection rates. Our reflection focuses instead on the nature of the comments, the types of issues made visible, and the way the activity supported discussion during refinement.

A second limitation concerns the role of the mentor and the collaborative nature of the activity. As discussed in Lesson 3, the mentor prepared and facilitated the training, observed the group inspections, supported the debriefing moments, and helped consolidate questions before the discussion with the PO. This role was central to the pedagogical design, but it may also have influenced participants' behavior and interpretation. The inspections were also conducted in groups, which was aligned with the agile and collaborative nature of the training program. Group discussion, however, can create interdependencies: some participants may have influenced the reasoning of others, and some comments may reflect group consensus rather than individual understanding. The findings should therefore be interpreted at the level of the team experience, not as evidence of individual learning or individual ability to inspect BDD scenarios.

Finally, the generalization of this experience is limited and the data cannot be fully shared. The participants were early-career developers in a specific corporate training program, working under its routines, constraints, and support structure, so the findings may not transfer directly to more experienced teams, different domains, or contexts in which the PO is less available during the sprint. The experience is most relevant to educators and training programs that work with early-career developers and need lightweight ways to support understanding of requirements and scenario refinement. In addition, because the scenarios belonged to a real internal project from ORG, the complete set of inspected scenarios cannot be disclosed. To support transparency, we report the structure of the activity, the types of artifacts used, the attributes adopted to organize the comments, and selected adapted examples that preserve the patterns observed during the experience without revealing sensitive project information.

7 Conclusion

This paper reported an experience of using a checklist of guiding questions to support early-career developers in the refinement of BDD scenarios during a corporate training program. The activity was integrated into regular sprint rituals across a edition of the program, with 20 interns. The participants worked with real scenarios written by a Product Owner for an internal software project.

By triangulating participants' self-reported perceptions with the refinement comments produced during the sprints, the results suggest that the checklist supported a more structured discussion of requirements quality. Without the checklist, participants identified relevant problems, but their comments were often uneven across groups and expressed as general doubts. With the checklist, comments became more explicitly connected to quality attributes and to different levels of the BDD artifact, including feature, scenario, and steps. This pattern suggests that structured guidance may help early-career developers notice and discuss issues such as missing scenarios, unclear triggers, vague wording, and weak alignment between scenario intention and expected behavior.

For Software Engineering education, this experience suggests that lightweight instruments such as checklists can be used not only as quality control artifacts, but also as pedagogical resources for supporting requirements understanding in agile training contexts. Future work may replicate this experience in other educational and onboarding settings, involve Product Owners and requirements engineers in extending the checklist, and investigate its use over longer observation periods.

Artifact Availability

Due to a non-disclosure agreement with the partner organization, the full set of inspected BDD scenarios cannot be made publicly available. Therefore, selected examples were prepared and included only for illustrative purposes.

Ethical Considerations on the Use of AI

We declare that Generative Artificial Intelligence tools were used in this work as follows: ChatGPT (OpenAI) assisted with the revision of English spelling and paragraph clarity during manuscript writing and editing, additionally, AI tools were used to enhance Figures 2, and 3. All generated content was reviewed and validated by the authors, who assume full responsibility for the originality and accuracy of the final text, in accordance with CNPq Ordinance No. 2.664/2026.

Acknowledgements

Natalya Goelzer thank Dell Brazil using incentives of the Brazilian Informatics Law (Law no. 8.2.48, year 1991) for their financial grants. Sabrina Marczak thanks CNPq (304795/2024-0).

References

[1] Gojko Adzic. 2009. *Specification by Example: How Successful Teams Deliver the Right Software.* Manning Publications.

[2] Giovanna C. S. Bettin, Ricardo Theis Geraldi, and Edson OliveiraJr. 2018. Experimental Evaluation of the SMartyCheck Techinique for Inspecting Defects in UML Component Diagrams. In *XVII Brazilian Symposium on Software Quality* (Curitiba, Brazil) *(SBQS '18)*. Association for Computing Machinery, New York, NY, USA, 101–110. doi:10.1145/3275245.3275256

[3] Jardelane Brito and Arilo Claudio Dias Neto. 2013. TestCheck – Uma Abordagem Baseada em Checklist para Inspecionar Artefatos de Teste de Software. In *Anais do XII Simpósio Brasileiro de Qualidade de Software* (Salvador). SBC, Porto Alegre, RS, Brasil, 366–380. doi:10.5753/sbqs.2013.15300

[4] Nayana Carneiro, Bianca Melo, Letícia Cavalcante, Rute Castro, Rossana M. C. Andrade, and Ticianne Darin. 2021. Mobili: Development and Use of a Usability Checklist for Mobile Games and Applications. In *Proceedings of the XIX Brazilian Symposium on Software Quality* (São Luís, Brazil) *(SBQS '20)*. Association for Computing Machinery, New York, NY, USA, Article 33, 9 pages. doi:10.1145/3439961.3439994

[5] Thiciane Couto, Sabrina Marczak, Daniel Callegari, Michael Móra, and Fabio Gomes. 2022. On the Characterization of Behavior-Driven Development Adoption Benefits: A Multiple Case Study. In *Anais do XXI Simpósio Brasileiro de Qualidade de Software* (Curitiba/PR). SBC, Porto Alegre, RS, Brasil, 150–159. https://sol.sbc.org.br/index.php/sbqs/article/view/23301

[6] Thiago da Cunha, Ismayle Santos, Alysson de Macedo, Alberto Monteiro, Bruno Aragão, and Rossana Andrade. 2016. CAS 2.0: Evolução e Automação do Checklist de Avaliação do Scrum para Projetos de Software. In *Anais do XV Simpósio Brasileiro de Qualidade de Software* (Maceió). SBC, Porto Alegre, RS, Brasil, 106–120. doi:10.5753/sbqs.2016.15129

[7] Rafael Maiani de Mello, Rebeca Campos Motta, and Guilherme Horta Travassos. 2016. A checklist-based inspection technique for business process models. In *Business Process Management Forum: BPM Forum 2016, Rio de Janeiro, Brazil, September 18-22, 2016, Proceedings 14*. Springer, 108–123. doi:doi:10.1007/978-3-319-45468-9_7

[8] Rafael Maiani de Mello, Eldânae Nogueira, Marcelo Schots, Cláudia Maria Lima Werner, and Guilherme Horta Travassos. 2014. Verification of Software Product Line Artefacts: A Checklist to Support Feature Model Inspections. *Journal of Universal Computer Science* 20, 5 (2014), 720–745. doi:doi:10.3217/jucs-020-05-0720

[9] Michael E. Fagan. 1976. Design and code inspections to reduce errors in program development. *IBM Systems Journal* 15, 3 (1976), 182–211.

[10] Muhammad Shoaib Farooq, Uzma Omer, Amna Ramzan, Mansoor Ahmad Rasheed, and Zabihullah Atal. 2023. Behavior Driven Development: A Systematic Literature Review. *IEEE Access* 11 (2023), 89274–89298. doi:10.1109/ACCESS.2023.3302356

[11] Atul Gawande. 2009. *The Checklist Manifesto: How to Get Things Right.* Metropolitan Books.

[12] Gonzalo Génova, José M. Fuentes, Juan Llorens, Omar Hurtado, and Valentín Moreno. 2013. A framework to measure and improve the quality of textual requirements. *Requir. Eng.* 18, 1 (March 2013), 25–41. doi:10.1007/s00766-011-0134-z

[13] Natalya Goelzer and Sabrina Marczak. 2024. Promoting Quality in BDD Scenarios Using Checklist: An Investigation from the Perspective of Novice Professionals. In *Anais do XXIII Simpósio Brasileiro de Qualidade de Software* (Bahia/BA). SBC, Porto Alegre, RS, Brasil, 341–350. https://sol.sbc.org.br/index.php/sbqs/article/view/32961

[14] Natalya Goelzer, Pedro Possamai, Sabrina Marczak, Michael Móra, and Daniel Callegari. 2024. Nurturing Talent: The IT Academy Journey into Quality Development. In *Anais do XXIII Simpósio Brasileiro de Qualidade de Software* (Bahia/BA). SBC, Porto Alegre, RS, Brasil, 508–518. https://sol.sbc.org.br/index.php/sbqs/article/view/32978

[15] Mohsin Irshad, Jürgen Börstler, and Kai Petersen. [n. d.]. Supporting refactoring of BDD specifications—An empirical study. *Information and Soft. Tech.* ([n. d.]). doi:doi:10.1016/j.infsof.2021.106717

[16] Adriana Lopes, Ursula Campos, Tayana Uchoa Conte, and Clarisse Sieckenius de Souza. 2018. ComD2: Family of Techniques for Inspecting Defects in Models that Affect Team Communication. In *30th International Conference on Software Engineering and Knowledge Engineering* (Pittsburgh, PA). 298–303.

[17] Sômulo Mafra and Guilherme Travassos. 2005. Técnicas de Leitura de Software: Uma Revisão Sistemática. In *Anais do XIX Simpósio Brasileiro de Engenharia de Software* (Uberlândia/MG). SBC, Porto Alegre, RS, Brasil, 72–87. doi:10.5753/sbes.2005.23812

[18] Nicolas Nascimento, Alan R. Santos, Afonso Sales, and Rafael Chanin. 2020. Behavior-Driven Development: A case study on its impacts on agile development teams. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (Seoul, Republic of Korea) *(ICSEW'20)*. Association for Computing Machinery, New York, NY, USA, 109–116. doi:10.1145/3387940.3391480

[19] Christiano Neitzke, João Mendes, Luis Rivero, Mario Teixeira, and Davi Viana. 2023. Enhancing LGPD Compliance: Evaluating a Checklist for LGPD Quality Attributes within a Government Office. In *Proceedings of the XXII Brazilian Symposium on Software Quality* (Brasília, Brazil) *(SBQS '23)*. Association for Computing Machinery, New York, NY, USA, 218–227. doi:10.1145/3629479.3629497

[20] Gabriel Oliveira. 2018. *Question-based checklist to evaluate BDD scenarios' quality.* Master's thesis. Faculdade de Informática - PUCRS. Escola Politécnica.

[21] Gabriel Oliveira and Sabrina Marczak. 2018. On the Understanding of BDD Scenarios' Quality: Preliminary Practitioners' Opinions. *IEEE 25th International Requirements Engineering Conference Workshops, Cham, Switzerland, Suiça* (2018), 290–296. doi:10.1007/978-3-319-77243-1_18

[22] Gabriel Oliveira, Sabrina Marczak, and Cassiano Moralles. 2019. How to Evaluate BDD Scenarios' Quality?. In *Anais do XXXIII Simpósio Brasileiro de Engenharia de Software, Salvador, Brasil* (Salvador, Brazil). Association for Computing Machinery, New York, NY, USA, 481–490. doi:10.1145/3350768.3351301

[23] Lauriane Pereira, Helen Sharp, Cleidson de Souza, Gabriel Oliveira, Sabrina Marczak, and Ricardo Bastos. 2018. Behavior-Driven Development Benefits and Challenges: Reports from an Industrial Study. *19th International Conference on Agile Software Development* 18, 42 (2018), 4. doi:10.1145/3234152.3234167

[24] F. Shull, I. Rus, and V. Basili. 2000. How perspective-based reading can improve requirements inspections. *Computer* 33, 7 (2000), 73–79. doi:10.1109/2.869376

[25] John Smart. 2014. *BDD in Action: Behavior-Driven Development for the Whole Software Lifecycle.* Manning Publications, Shelter Island, NY.

[26] Matt Wynne and Aslak Hellesoy. 2012. *The Cucumber Book: Behaviour-Driven Development for Testers and Developers.* Pragmatic Bookshelf.