IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 10, 2020, accepted March 30, 2020, date of publication April 20, 2020, date of current version May 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2989106

A Catalogue of Agile Smells for Agility Assessment

ULISSES TELEMACO[1], TOACY OLIVEIRA[1], PAULO ALENCAR[2], AND DON COWAN[2]
[1]System Engineering and Computing Science Program, Federal University of Rio de Janeiro, Rio de Janeiro 21941-914, Brazil
[2]David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada

Corresponding author: Ulisses Telemaco (ulisses.telemaco@owse.com.br)

This work was supported by the Advanced High Performance Computing Center (NACAD/COPPE) at the Federal University of Rio de Janeiro (UFRJ) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

ABSTRACT **Background:** The Manifesto for Agile Development has already inspired many software development methods such as Scrum, XP, and Crystal Reports. However, being ''agile'' is not trivial and only a few companies are capable of mastering so-called agile practices. Failure to apply the agile approach properly can do more harm than good and may jeopardize the benefits of an agile method. Thus, evaluating an organization's ability to apply agile practices using an agility assessment tool is critical. **Aims:** In this paper, we extend the metaphor of code smell and introduce the term agile smell to denote the issues and practices that may impair the adoption of the agile approach. The focus of the paper is defining and validating a catalogue of agile smells that can support agility assessment. **Method:** A literature review and a survey were conducted to identify and confirm the characterization of agile smells. Once identified, the agile smells were organized in a structured catalogue. **Results:** The literature review found 2376 references published between 2001 and 2018. We selected 55 papers for full consideration and identified 20 agile smells. The survey consulted 20 participants to determine the relevance of the selected agile smells. **Conclusion:** We have identified a set of 20 agile smells that were ranked according to their relevance. For each smell, we proposed at least one strategy to identify the smell's presence in real projects. The catalogue can be used by companies to support the assessment of their agility ability.

INDEX TERMS Agility assessment, agile development, agile smell.

I. INTRODUCTION

The adoption of agile methods by the software development industry has increased significantly in recent years. Almost all software companies claim they are ''*agile*'' at some level and they are using agile practices in their software processes [1]. Being *agile* has become a critical factor for the Software Industry. Among the expected benefits of being agile are the acceleration of software delivery through the ability to manage requirement changes and productivity increments [2].

However, the proper adoption of an agile method (or agile practices) is not straightforward and the misuse of agile practices should not be ignored since it may jeopardize the benefits that an agile method should bring to the organization. It is quite common to find organizations new to agile software development techniques, adopt a few agile practices, adapt them in the way they prefer and convince themselves they are doing agile software development until they eventually realize there are no or few improvements in their software processes [3]. Ambler [4] revealed numerous project failures associated with agile development. In the *2018 IT Project Success Rates Survey*™, 36% of the participants reported that they had experienced challenges in an agile project, and 3% of the participants reported complete failure [5].

Thus, an Agility Assessment (AA) tool is a critical approach to assist projects, organizations and even individuals in understanding their agility skills and identifying potential problems that should be resolved to improve the adoption of agile methods [6].

The problem we have observed is the lack of an objective criteria for conducting an agility assessment. Despite the substantial amount of content about agile development in both academic forums and industry, there are few contributions that focus on providing elements to support agility assessment. The *Manifesto for Agile Development* [7], for example, proposed a set of values and principles that have inspired many agile methods. However, using these values

The associate editor coordinating the review of this manuscript and approving it for publication was Resul Das.

and principles as a base to assess the agility of a given organization or project is quite difficult. It is challenging and subjective to assess whether an organization or project is properly applying values such as the requirement to focus on *"individuals and interactions over processes and tools"*. Agile methods such as Scrum [8], XP [9], Crystal Family Methods [10] and Open Up [11] or other studies that consolidated the body of knowledge around agile development do not provide objective requirements for assessing the adoption of agile practices. The so-called agile values, principles, practices and characteristics are typically described: (a) in a generic way; (b) to be used as reference for projects or organizations that aim adopting agile, or (c) to inspire discussions among the team in retrospective meetings.

Agility assessment approaches need objective criteria otherwise the assessment may be threatened by biases imposed by the person(s) conducting the assessment. This paper tries to fill this gap by proposing a set of practices focused on agility assessment. We borrowed the term *code smell* [12] and extended it to agility assessment. A *code smell* denotes an indication that may correspond to a deeper problem in the software source code or architecture. The term was popularized by Fowler and Beck in [12]. The authors used this metaphor and proposed a catalogue of code smells that can be used to guide the identification of potential problems that could be fixed through the application of refactoring techniques. We are using the term *agile smell* to denote a practice that may impair the proper adoption of agile development.

This paper aims at identifying a set of agile smells and organizing them in a structured format. We are also proposing, for each agile smell, at least one strategy that guides the identification of the occurrence of that agile smell in an agile project. The methodology of this study is divided into three phases: (a) an elicitation phase that includes a literature review; (b) a confirmation phase that includes a survey with practitioners; and (c) a cataloging phase that attempts to organize the agile smells in a structured format.

This study tries to answer the following research questions:

RQ1: *What are the practices that impair the proper adoption of agile development and can be used to support the agility assessment of organizations, projects, iterations and agile teams?*

RQ2: *How can we identify the occurrence of such practices?*

The aim of RQ1 is to identify a set of items that we are naming *agile smells*, which are: practices that may jeopardize the adoption of agile development and that can also be used to support organizations and agile teams to assess how they are using agile practices. To answer RQ1, we are proposing a catalogue of agile smells that were identified through a literature review and confirmed by a survey. The aim of RQ2 is to propose strategies to identify the occurrences of agile smells. An identification strategy is important to make agility assessment less subjective and less compromised by evaluator bias. These strategies will aid and guide practitioners to quickly spot the occurrence of agile smells in an agile project. We sought to answer RQ2 by proposing at least one identification strategy for each agile smell. By answering these two questions, we expect to provide a baseline to support agility assessment at organizational and project levels.

An early version of this study was introduced in [13], which presented a preliminary version of the catalogue and a small set of agile smells. We have improved on the study in [13] by:

- broadening the literature review;
- expanding the number of participants in the survey;
- consolidating the set of agile smells;
- adding more information into the catalogue; and
- adding the catalogue use guideline.

The remainder of this paper is organized using the following structure: Section II presents the background for this research. Section III describes the study methodology. Sections IV and V describe, respectively, the literature review and the survey conducted to identify and confirm the agile smells. Section VI describes the catalogue design and presents a subset of the resulting catalogue. The catalogue use guideline is introduced in Section VII. Section VIII discusses the related work and Section IX presents the results and the threats to the validity of this study. Section X concludes the paper.

II. BACKGROUND

In this section, concepts that include Agile Development, Agility Assessment and Code Smells are discussed to provide background material for the reader.

A. AGILE DEVELOPMENT

In 2001, as a response to a community that demanded more flexible processes, a group of 11 practitioners and consultants in software development produced what they named *Manifesto for Agile Development* [7]. The values and principles that were the foundation of the manifesto and that were proposed as an attempt to influence the software development community are presented in the following four values and 12 underlying principles in Figure 1:

While there is no formal agreement on the meaning of the concept of "agile", in this research, "*Agile Development*" means software development processes or methods that are shaped around the values and principles in Figure 1. These methods include, but are not limited to: *XP* [9], [14], [15], *Scrum* [8], [16]–[18], *Crystal Family* [10], *Feature Driven Development (FDD)* [19], [20], *Dynamic Systems Development Method* [21], *Adaptive Software Development* [22], and *OpenUp* [23].

B. AGILITY ASSESSMENT

The adoption of so-called agile practices may not be straightforward [24]–[26]. The *13th Annual State of Agile Survey*^{TM} [1] revealed that, although 94% of companies

Value 1: ***Individuals and interactions*** *over processes and tools*;
Value 2: ***Working software*** *over comprehensive documentation*;
Value 3: ***Customer collaboration*** *over contract negotiation*; and
Value 4: ***Responding to change*** *over following a plan.*

Principle 1: *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software*;
Principle 2: *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage*;
Principle 3: *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale*;
Principle 4: *Business people and developers must work together daily throughout the project*;
Principle 5: *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done*;
Principle 6: *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation*;
Principle 7: *Working software is the primary measure of progress*;
Principle 8: *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely*;
Principle 9: *Continuous attention to technical excellence and good design enhances agility*;
Principle 10: *Simplicity (the art of maximizing the amount of work not done) is essential*;
Principle 11: *The best architectures, requirements, and designs emerge from self-organizing teams*; and
Principle 12: *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*

FIGURE 1. **Agile values and principles.**

surveyed claimed they are using agile practices, only 4% of the companies indicated they are mastering agile practices. In this scenario, it is important for organizations to identify their gaps in agile practices, otherwise, the organization may not receive the benefits of adopting them [4].

Agility Assessment (AA) comprises assessment techniques, models and tools that focus on indicating problems in adopting agile practices at a project-level, organization-level or individual-level. There are many approaches for AA such as agility assessment models, agility checklists, agility surveys, and agility assessment tools [27]. In Section VIII we present some of these AA approaches and discuss how they relate to this study.

C. CODE SMELLS AND AGILE SMELLS

The term *code smell* was popularized by Fowler and Becker [12] to describe poor design solutions and code structures that should be analyzed carefully. The authors proposed a baseline catalogue of code smells that is divided in three categories (Application-level, Class-level and Method-level) and includes 22 smells such as *Duplicated Code* (identical or very similar code exists in more than one location), *Shotgun Surgery* (a single change needs to be applied to multiple classes at the same time), *Large Class* (a class that has grown too large), *Feature Envy* (a class that uses methods of another class excessively), *Inappropriate Intimacy* (a class that has dependencies on implementation details of another class), *Cyclomatic Complexity* (too many branches or loops), *Too Many Parameters* (a long list of parameters), and *Long Method* (a method, function, or procedure that has grown too large).

In this study, we extend the term *Code Smell* to the context of agile development and propose the term *Agile Smell*. An *Agile Smell* denotes a practice likely to impair the proper adoption of agile development.

D. AGILITY ASSESSMENT AND AGILE SMELLS

The catalogue of code smells [12] proposed by Fowler and Becker and other contributions [28]–[30] have been broadly used by the software industry to assess the quality of their source-code. However, manually identifying the occurrence of code smells in projects that could have hundreds of thousands of code lines is costly and neither effective nor efficient and a more scalable technique is needed [31], [32]. One of the approaches to optimize the source-code quality assessment is through automatically detecting code smells [29], [32]–[34]. These techniques require the specification of a code smell in a specific language. The DECOR [32] method proposed by Moha *et al.*, for example, is organized in four steps: *Description Analysis*, *Specification*, *Processing*, *Detection*, and *Validation*. In the *Specification* step, the smells are coded in a specification language. These specifications are then used as input for the *Detection* step that assesses the code and finds potential code smells. The code smells identified in this step are confirmed in the *Validation* step.

In this sense, the catalog of agile smells proposed in this study could be used to guide agility assessments and be a prelude for automatic detection of agile smells.

III. STUDY ORGANIZATION

This section presents the research methodology followed to identify and confirm a set of agile practices that may impair the adoption of agile methods (AKA agile smells). The methodology of this research was based on the method proposed by Spinola et al [35] and consists of four steps divided into three phases as depicted in Figure 2:

Phase 1 - Elicitation: The first phase, elicitation phase, was divided in two steps: (1) An informal literature review that was conducted to identify basic concepts that supported the definition of an accurate and comprehensive systematic literature review protocol; and (2) A systematic literature review that was planned and executed to identify a set of agile smells. The systematic literature review design details, the mechanisms and collected data and the set of identified agile smells are described in Section IV.

Phase 2 - Confirmation: In the confirmation phase, we conducted a survey with industry practitioners to confirm the agile smells identified in the elicitation phase and reveal their relevance. The survey is described in Section V

Phase 3 - Consolidation: In this phase, the most relevant agile smells were organized in a structured format named

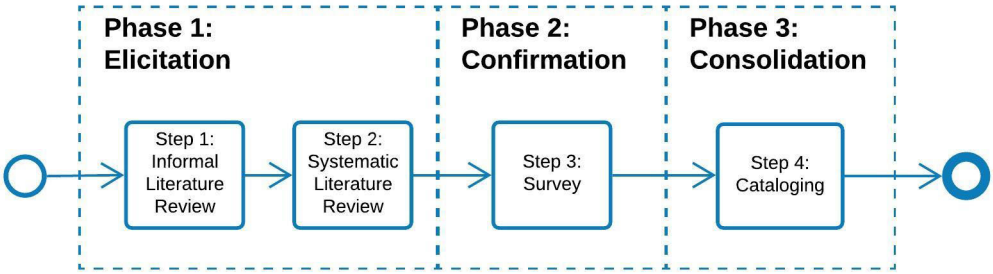


FIGURE 2. The research methodology organized in three phases and four steps.

the *Catalogue of Agile Smells*. This catalogue is presented in Section VI.

IV. ELICITATION PHASE: SYSTEMATIC LITERATURE REVIEW

In the elicitation phase, a systematic literature review (SLR) was conducted to explore the existing body of knowledge and identify a set of agile smells (ie. practices that may impair the proper adoption of agile development).

The methodology of the SLR was based on the method proposed by Kitchenham and Charters [36] and consists of three main phases: *planning*, *execution* and *reporting*.

A. SYSTEMATIC LITERATURE REVIEW PLANNING

1) AIM, RESEARCH QUESTIONS AND SCOPE

The aim of the literature review is to identify elements that allow us to answer the RQ1 and RQ2 questions. In other words, the goal of the SLR is to discover (i) a set of practices that may impair the adoption of agile development (AKA agile smells) and (ii) strategies on how to check for the occurrence of these practices in real projects. Since the literature does not use the term ''Agile Smell'', we extracted the agile smells from agile practices, rules, constraints or restrictions. The research questions for this SLR were derived from the RQ1 and RQ2 (presented in Section I) and can be summarized as:

SLR-RQ1: *What are the practices that impair the proper adoption of agile development?*
SLR-RQ2: *How can we identify the occurrence of such practices?*

The scope of this review was defined based on the population, intervention, comparison and outcome (PICO [37]) approach. The *Population* is the set of software development projects. The *Intervention* is the collection of agile software development processes. There is no comparison. The outcome is a set of agile rules, constraints, practices and techniques. Three papers obtained from a previous conventional literature review were used as control:

1) [38] Miller, G.G.: The characteristics of agile software processes. In: Proceedings of the 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS39). TOOLS '01, IEEE Computer Society, Washington, DC, USA (2001)
2) [39] Lindvall, M., Basili, V.R., Boehm, B.W., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams, L.A., Zelkowitz, M.V.: Empirical findings in agile methods. In: Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002. Springer-Verlag, London, UK, UK (2002)
3) [40] Abrantes, J.F., Travassos, G.H.: Common agile practices in software processes. In: 2011 International Symposium on Empirical Software Engineering and Measurement. pp. 355–358 (Sept 2011) .

The keywords for *Population* are ''software process'', ''software projects'', ''software systems'', ''software development'', and ''software engineering''. The keywords for *Intervention* are ''agile methods'', ''agile processes'', ''agile approaches'', ''agile methodologies'', and ''agile development''. The keywords for *Outcome* are ''rules'', ''constraints'', ''restrictions'', ''practices'', ''technics/techniques'', and ''classification''. The sources are collected from the following digital databases, including conferences, journals and technical reports indexed by ACM Digital Library, IEEE Xplore, Scopus, and Web of Science. The search string taken as the basis for all search engines, structured according to Pai *et al.* [37] was: *(''software process'' or ''software project'' or ''software systems'' or ''software development'' or ''software engineering'') and (''agile methods'' or ''agile processes'' or ''agile approaches'' or ''agile methodologies'' or ''agile development'') and (''rules'' or ''constraints'' or ''restrictions'' or ''practices'' or ''technics'' or ''techniques'' or ''classification'')*

The set of formal literature studies includes all articles returned by the protocol that meets at least one of the following inclusion criteria (IC): (IC1) Documents must address one or more agile methods; (IC2) Documents must discuss practices, characteristics, rules or constraints related to an agile method.

Publications that satisfy at least one of the following exclusion criteria (EC) were omitted: (EC1) Documents not written in English; (EC2) Documents whose full text is not available; (EC3) Documents clearly dealing with topics irrelevant to

the purpose of this review; (EC4) Documents merely reporting the use of individual software processes in development projects; (EC5) If the same study has been published more than once, the most relevant version, such as the one explaining the study in greatest detail will be used and the others will be excluded.

2) DATA EXTRACTION CRITERIA

To identify and extract the agile smells from the selected studies, we defined two data extraction criteria (DEC): (DEC1) an agile smell is a practice that may impair the adoption of agile methods; and (DEC2) the occurrence of an agile smell should be objectively verified. The DEC1 criterion defines an agile smell as a negative practice that should be avoided. The DEC2 criterion was introduced to reduce the risk of identifying agile smells that are vague or hard to be verified through objective strategies. Note that the gap this research is trying to fulfill is the lack of objective criteria to perform an agility assessment. The values and principles proposed by the *Manifesto for Agile Development* and the methodologies derived from the manifesto are described in a vague way [41], [42]. Therefore, identifying agile smells that are difficult to be objectively verified would not differentiate them from the body of knowledge already consolidated in this area.

The following information was extracted from each paper selected after running the data extraction process: *document title*, *author(s)*, *source*, *year of publication*, *agile method*, *agile smell name* and *agile smell description*. The results were tabulated. Analysis was carried out to identify duplication.

B. SYSTEMATIC LITERATURE REVIEW EXECUTION

After the planning phase, seven steps were applied in the execution phase to select the primary studies:

- *Step 1: Initial Search.* We applied the search string to the selected digital databases. A broad number of studies was retrieved in this phase: ACM Digital Library (438), IEEE Xplore (564), Scopus (2233), and Web of Science (1592).
- *Step 2: Combination.* Since the digital databases index many of the same publications [43], we combined the results and the total number of studies after this step was 2376. All the control studies were retrieved.
- *Step 3: Filter by Title.* This step aimed at applying the exclusion criteria EC1, EC2, EC3 and EC4 by reading the title of the studies. After this step, the number of papers was reduced to 261.
- *Step 4: Filter by Abstract.* This step aimed at applying the exclusion criteria EC3 and EC4 by reading the abstract of the studies. At the end of this step, 127 studies remained.
- *Step 5: Filter by full text.* It consisted of filtering the selected studies by reading their full text and applying the exclusion criteria EC3 and EC4. At the end of this step, 42 studies remained.
- *Step 6: Removal of repeated studies.* We applied the exclusion criterion EC5 and removed two studies. After this step, the number of papers selected for full consideration was reduced to 40.
- *Step 7: Addition by Heuristic.* We inserted 15 relevant studies from other sources, totaling 55 studies. These studies were added manually, based on our background knowledge. Appendix X shows the final list of studies considered in this literature review.

Figure 3 shows the process and the results obtained in each step. The selected documents were fully read and the data extraction criteria applied to identify the agile smells.

C. LITERATURE REVIEW REPORTING

During the SLR, we identified many agile values, practices and characteristics. However, none of the studies investigated agile methods from the perspective of this study, namely, trying to identify a set of agile practices that may impair the adoption of agile methods. The SLR confirmed that most of the body of knowledge around agile development focused on adoption of agile development rather than agility assessment. The studies neglected to describe explicitly how to verify whether the values, practices and characteristics of agile development have been properly adopted.

After reading the selected papers, we extracted 20 agile smells using the two data extraction criteria (DEC1 and DEC2) established in the research protocol. The identified agile smells answer research question SLR-RQ1 and are presented below in alphabetical order:

1) **Absence of Frequent Deliveries**: The practice of delivering products continuously and frequently is very important to agile methods and that is almost a mantra among agile software developers. The *Absence of Frequent Deliveries* smell is detected when the development team does not deliver a new version of the software frequently. The occurrence of this smell may indicate that this practice has been jeopardized. References: [8]–[10], [14]–[23], [40], [44]–[53].
2) **Absence of Test-driven Development**: Test Driven Development (TDD) is an agile software development technique that is based on the following short cycle of repetitions: First, the developer writes a test case that defines the desired behavior for a new functionality. Then, the code is written that can be validated by the test case. The *Absence of Test-driven Development* smell is detected when the development team does not apply the technique TDD (Test Driven Development) during the development of the software. The presence of this smell may indicate the team is not applying the TDD technique. References: [2], [9], [10], [14], [15], [21], [40], [44], [45], [47]–[49], [52], [54]–[61].
3) **Absence of Timeboxed Iteration**: The *Timeboxed Iteration* practice defines that all iterations should have a fixed time duration. Thus, an iteration should not be extended or shortened to fit planned or

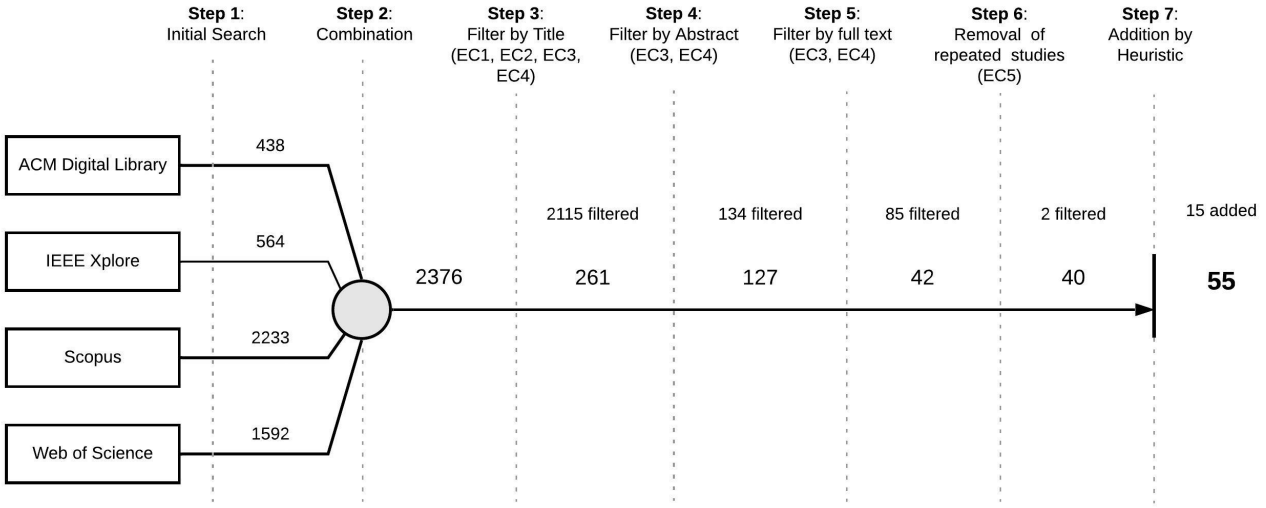


FIGURE 3. The process of primary study selection.

unplanned features. The *Absence of Timeboxed Iteration* smell is detected when an iteration is shorter or longer than the predefined duration. The presence of this smell may indicate the timeboxed iteration practice has not been applied properly. References: [8], [10], [16]–[18], [22], [38], [46], [48]–[50], [53]–[55].

4) **Absence of Timeboxed Meeting**: This smell derives from an agile practice that states the meetings prescribed by the agile method (iteration planning, review, retrospective, etc) should have a predefined duration and the duration should preferably be the same during the entire software project. The *Absence of Timeboxed Meeting* smell is detected when a given meeting (prescribed by the agile method) is shorter or longer than the predefined duration. The presence of this smell may indicate the team is not properly conducting the meeting or they are not planning the meetings properly. References: [18], [44], [46], [48], [54]. [52], [57].
5) **Complex Tasks**: Complex tasks should be avoided in agile projects. They should be decomposed by the development team into simpler tasks. The *Complex Tasks* smell is detected when there are complex tasks in a given iteration. The presence of this smell may indicate that the developers are not properly breaking complex tasks into simpler tasks. References: [8], [16], [17], [40], [46], [47], [49], [52], [55], [61], [62].
6) **Concurrent Iterations**: In an agile project, the entire team should focus on the same iteration goal. Running two (or more) consecutive iterations means the team is divided and focused on different goals. The *Concurrent Iterations* smell is detected when there are two (or more) open iterations in the same project. The presence of this smell may indicate the development team is not focused on the same goal. References: [18], [55], [63], [64].
7) **Dependence on Internal Specialists**: One characteristic of an ideal agile team is one in which any participant can work on any feature. Thus, the team should avoid the situation where a member becomes the only specialist in a feature or technology. The *Dependence on Internal Specialists* smell is detected when all tasks related to a given feature were assigned to the same developer. The presence of this smell may indicate the creation of an internal specialist and the project is becoming dependent on a specific developer. References: [2], [8], [9], [14], [15], [17], [18], [44], [45], [47]–[50], [54], [55], [65]–[69].
8) **Goals Not Defined**: Agile development teams need to know exactly what they are working on and the goals of the project and iterations should be clear and well-defined. The *Goals Not Defined* smell is detected when the goals of the project or of a given iteration are not defined. The presence of this smell may indicate the development team does not have a clear view of the goals and therefore could not choose the most important work to do. References: [8], [10], [16], [17], [19]–[23], [38], [46], [48], [51], [69]. [52].
9) **Iteration Started without an Estimated Effort**: The scope and duration of the iterations in an agile project are typically defined by the development team that must commit to the iteration goals and deadlines. The *Iteration Started without an Estimated Effort* smell is detected when an iteration that contains non-estimated tasks is started. The presence of this smell may indicate that the development team is committed to a deadline without a good understanding of the effort to deliver the iteration scope. References: [18], [61], [63], [64], [67], [69].
10) **Iteration Without a Deliverable**: The practice of delivering products continuously and frequently is very important to agile methods and can be considered a mantra among agile software developers. The agile methods state the development team should deliver a

new version of the software at the end of each iteration. The *Iteration Without a Deliverable* smell is detected when an iteration does not have an associated deliverable product. The presence of this smell may indicate that the continuous and frequent delivery practice has been jeopardized. References: [8]–[10], [14]–[23], [40], [44]–[46], [48], [49], [51], [53].

11) **Iteration Without an Iteration Planning**: Iteration planning is an important success factor in agile methods. Normally an iteration plan is elaborated with the main stakeholders (developers and customer) that together decide what should be developed in the iteration. The *Iteration Without an Iteration Planning* smell is detected when there is no planning associated with a given iteration. The presence of this smell may indicate that the iterations are not being planned properly. References: [8]–[10], [14]–[23], [40], [44]–[50], [52], [54]–[61], [64], [66], [67], [70], [71].
12) **Iteration Without an Iteration Retrospective**: Retrospective meetings represent opportunities for the development team to reflect on how they are working and improve the method when necessary. The *Iteration Without an Iteration Retrospective* smell is detected when there is no retrospective meeting associated with a given iteration. The presence of this smell may indicate that an important opportunity for improvement prescribed by agile methods is being wasted. References: [10], [16]–[23], [46]–[61].
13) **Iteration Without an Iteration Review**: The iteration review is a meeting where the development team presents to the product owner what was accomplished during the previous iteration. Typically, there is a software demonstration showing the new features and a discussion of what is being delivered. The *Iteration Without an Iteration Review* smell is detected when there is no review associated with a given iteration. The presence of this smell may indicate the development team is missing an important opportunity to present the results of the iteration to the product owner. References: [8], [10], [16]–[23], [46]–[49], [52]–[56], [58]–[60], [71].
14) **Iterations with Different Duration**: In order to promote sustainable development and to understand their productivity, the development team should work at a constant pace. That means the iterations in a given project should ideally have the same duration. The *Iterations with Different Duration* smell is detected when iterations in the same project do not have the same duration. The presence of this smell may indicate the development team is not maintaining a constant pace. References: [18], [47], [52], [55], [63], [64].
15) **Large Development Team**: An agile development team should be small to be efficient and effective. The *Large Development Team* smell is detected when the development team is larger than the predefined recommended size. References: [8], [16]–[20], [39], [48]–[50], [52], [53], [65], [67].
16) **Long Break Between Iterations**: To promote sustainable development and understand its productivity, the development team must measure all the work done. Since the work done during the interval between iterations is typically not counted in productivity assessment, long breaks may impact the way the team measures its productivity. The *Long Break Between Iterations* smell is detected when there is a break between two consecutive iterations longer than a predefined and recommended size. The presence of this smell may indicate the development team is working on untraceable work that can harm the calculation of team productivity. References: [18], [46], [55], [57], [60], [63], [64].
17) **Lower Priority Tasks Executed First**: In an agile project, the development team should focus on higher priority tasks. The *Lower Priority Tasks Executed First* smell is detected when tasks with lower priority are executed before tasks with higher priority. The occurrence of this smell may indicate that the development team has not worked on the highest priority tasks. References: [8]–[10], [14]–[17], [19]–[23], [44]–[46], [48], [49], [51]–[53], [55], [60], [69].
18) **Shared Developers**: In an agile project, business people and developers must work together daily throughout the project. Developers are expected to become experts in the project scope and switching a developer across multiple projects does not contribute to the involvement of that developer in the project. The *Shared Developers* smell is detected when a developer is working on more than one project at the same time or when that developer is frequently switching between different projects. The presence of this smell may indicate the organization is not properly allocating the developers. References: [2], [18], [46], [52], [53], [55], [61], [65].
19) **Unfinished Work in a Closed Iteration**: The entire scope of an iteration should preferably be delivered at the end of the iteration. But, as the iteration should be timeboxed, the development team must finish the iteration by the predefined deadline even if there is unfinished work. In that case, those unfinished work should be moved to the product backlog to be used in a future iteration planning. The *Unfinished Work in a Closed Iteration* smell is detected when a given iteration is closed even with unfinished tasks. The presence of this smell may indicate the team is not properly managing the backlog items and not moving unfinished work to the project backlog. References: [18], [46], [55], [63], [64]. [52], [69].
20) **Unplanned Work**: Agile teams usually commit to delivering a set of features before an iteration begins. To achieve the agreed commitment, the teams must work without interference, following the iteration plan and unplanned work should be avoided. The *Unplanned Work* smell is detected when tasks are included in a given iteration after it starts. The presence of this smell may indicate the unplanned tasks are jeopardizing the

commitment with the iteration deadline. References: [18], [44], [46], [48], [51], [52], [55], [60], [69].

To answer SLR-RQ2, we propose at least one identification strategy for each one of the agile smells identified in the literature review. For example, for the agile smell **Complex Tasks**, the following identification strategy was proposed:

1) **Identification Strategy for the Complex Tasks smell**: A strategy to identify the presence of the "Complex Tasks" smell is to verify whether the tasks estimates exceed an allowable threshold.

The identification strategies for other agile smells are presented in the catalogue in Section VI.

V. CONFIRMATION PHASE: SURVEY

In order to confirm the results from the literature review, we conducted a survey with practitioners based on semi-structured interviews [72]. The remainder of this section presents the survey that was based on the protocol proposed by Oishi [73]. The survey was divided into three phases: *planning*, *execution* and *reporting*.

A. SURVEY PLANNING

1) AIM AND RESEARCH QUESTIONS

The aim of the survey was to evaluate the relevance of the identified agile smells for an Agility Assessment. That is, how relevant is each of the agile smells to assess how an organization is using agile practices. The research questions for the survey are:

Survey-RQ1: *Is the given agile smell relevant to assess how an agile practice has been applied?*

Survey-RQ2: *Is the strategy for identification of the agile smell coherent and consistent with industry practices?*

2) INSTRUMENTATION AND QUESTIONNAIRE

The material used in the survey included an online questionnaire divided into three sections: (1) *Subject Characterization* (2) *Organization Characterization* and (3) *Agile Smells*. In *Subject Characterization* and *Organization Characterization* sections, the participants should provide information about themselves and the companies in which they work. The *Agile Smells* section contained a list of the 20 agile smells collected from the literature review. The agile smells were displayed in alphabetical order (as presented in Section IV-C) in the following structure: *Name*, *Short Description* and *Identification Strategy*.

SQ1: *What is the relevance of the given agile smell to assess how a project/organization is using agile practices?*

SQ2: *What is the relevance of the given identification strategy?*

$$relevanceByParticipant(p) = answerQuestion1(p) + answerQuestion2(p)$$

FIGURE 4. Formula of agile smell relevance by participant.

$$finalRelevance = \sum_{p=p1}^{pn} relevanceByParticipant(p)$$

FIGURE 5. Formula of final agile smell relevance.

TABLE 1. Summary of the survey participants characterization.

	# of participants	% of participants
Main Professional Occupation		
Project Manager	15	75.0%
Quality Assurance	5	25.0%
Highest Schooling Degree		
Associate	3	15.0%
Bachelor	7	35.0%
Master	6	30.0%
Doctoral	4	20.0%
Professional Experience		
5 to 10 years	5	25.0%
11 to 20 years	12	60.0%
> 21 years	3	15.0%
Geographic Distribution		
Brazil	16	80.0%
Canada	4	20.0%

The questionnaire accepted the following answers:

(a) *Not relevant (0 pts)*
(b) *Slightly relevant (1 pt)*
(c) *Very relevant (2 pts)*
(d) *Absolutely relevant (3 pts)*

Each answer has an associated value that varies from 0 to 3 (based on the relevance of the identification strategy) and that is used to calculate the relevance of the agile smell. The relevance of an agile smell, to a given participant, is the sum of the answers of Question 1 and Question 2 as shown in Fig. 4. Thus, to a given participant, the most relevant agile smell achieves a 6-point score and the least relevant agile smell has a 0-point score.

The final relevance of an agile smell is the sum of the relevance for all participants as illustrated in the formula presented in Fig. 5.

3) PARTICIPANTS SELECTION

We applied a convenience sampling approach [74] and participants were selected from our professional and academic networks. The criteria for the selection of participants were: (a) the participant should have at least 5-years experience as a Project Manager or Quality Assurance Consultant and (b) the participant should work or have worked in an organization that adopts a software process based on agile methods. We avoid selecting participants that are aware of this research, so we excluded coauthors and coworkers.

TABLE 2. Summary of data collected and analyzed in the survey.

Rank	Agile smell name	S1		S2		S3		S4		S5		S6		S7		S8		S9		S10		S11		S12		S13		S14		S15		S16		S17		S18		S19		S20		Total
01	Lower Priority Tasks Executed First	3	3	1	2	2	2	3	3	3	3	2	3	2	3	3	3	2	2	3	3	1	2	1	1	2	3	3	3	3	2	2	2	3	3	2	2	2	2	3	3	**96**
02	Absence of Frequent Deliveries	3	3	1	1	2	3	3	3	2	2	2	2	3	3	3	2	3	2	2	2	1	2	1	1	2	3	3	2	3	2	2	2	3	3	2	2	1	1	3	3	**90**
03	Iteration Without a Deliverable	2	2	2	3	2	2	2	2	3	2	2	2	3	2	3	3	1	2	2	3	1	1	1	1	2	2	2	2	3	3	3	2	3	3	2	2	3	3	1	1	**86**
04	Goals Not Defined	2	3	2	1	2	2	2	2	3	2	2	2	2	1	2	2	3	2	2	1	2	2	1	1	2	2	2	1	2	2	3	2	3	3	2	2	3	3	2	2	**82**
05	Iteration Without an Iteration Planning	2	2	2	1	2	2	2	1	2	3	2	2	2	2	1	2	1	2	2	2	3	3	1	1	2	1	2	2	3	2	2	2	3	3	2	2	3	3	2	2	**81**
06	Complex Tasks	2	3	1	1	3	2	2	2	2	2	1	1	2	1	2	2	2	1	2	1	2	1	2	2	2	1	3	2	3	3	2	2	3	3	2	2	2	2	2	2	**78**
07	Iteration Without an Iteration Retrospective	2	2	2	1	2	2	3	2	1	2	3	2	3	3	2	2	2	2	2	2	1	1	1	1	1	2	3	3	1	1	2	1	3	3	2	2	2	2	2	1	**77**
08	Absence of Timeboxed Iteration	2	3	1	2	2	2	2	2	3	2	2	3	1	1	2	1	2	2	2	1	2	1	1	1	1	2	2	2	1	1	2	2	3	3	2	2	2	2	3	3	**76**
09	Iteration Started without an Estimated Effort	2	2	2	1	1	1	2	1	2	2	1	1	1	1	1	1	1	2	2	2	3	2	1	1	2	1	3	3	3	3	2	2	3	3	2	2	3	3	2	2	**75**
10	Iteration Without an Iteration Review	1	2	2	1	2	1	1	2	2	2	2	1	1	1	2	1	2	2	2	2	2	1	1	1	2	2	3	3	2	1	2	2	3	3	2	2	3	3	2	2	**74**
11	Shared Developers	3	2	2	1	3	2	2	1	2	1	2	1	1	1	1	1	0	0	1	0	3	3	1	1	1	1	3	3	2	1	2	2	2	2	2	2	3	3	3	3	**70**
12	Unplanned Work	2	2	2	1	2	1	2	2	2	2	1	1	2	1	1	1	2	2	2	2	2	2	1	1	1	1	1	1	2	1	2	2	3	3	2	2	3	3	2	2	**70**
13	Dependence on Internal Specialists	2	1	2	1	2	1	2	1	2	2	1	2	1	1	2	1	2	2	2	2	1	1	1	1	2	1	3	3	1	1	1	2	3	3	2	2	3	3	1	2	**69**
14	Unfinished Work in a Closed Iteration	2	3	1	1	3	2	1	0	1	1	1	0	1	1	1	1	1	0	1	0	2	2	1	1	2	2	3	3	3	3	3	3	2	2	2	2	2	2	3	3	**67**
15	Absence of Timeboxed Meeting	2	2	2	1	1	1	2	2	2	2	1	1	1	1	1	1	2	2	2	2	2	2	1	1	2	1	2	2	1	2	2	1	2	2	2	2	1	1	2	2	**64**
16	Absence of Test-driven Development	2	2	1	1	1	1	2	1	1	1	1	0	1	1	0	0	2	1	2	2	1	1	1	1	2	1	3	3	0	0	3	3	3	3	2	2	2	2	3	3	**62**
17	Large Development Team	2	1	1	1	2	1	1	0	1	1	1	0	1	1	1	1	0	0	1	0	2	2	1	1	2	2	3	3	2	2	1	2	3	3	2	2	1	1	2	3	**57**
18	Long Break Between Iterations	1	1	1	0	1	1	2	1	1	0	1	0	1	1	0	0	1	1	1	1	2	2	1	1	1	1	3	3	2	2	1	2	3	3	2	2	3	3	2	2	**57**
19	Iterations with Different Duration	1	2	1	1	1	1	2	1	1	1	0	1	1	0	0	2	1	1	1	3	2	1	1	1	1	3	2	0	2	0	1	2	2	2	2	1	1	2	2		**51**
20	Concurrent Iterations	1	1	0	0	1	0	1	0	1	1	1	2	1	1	2	1	1	1	1	0	2	2	2	2	1	0	1	0	1	1	1	1	3	3	2	2	3	3	1	1	**49**

During the planning phase, we conducted a preliminary analysis using subjects from inside our research group. The data from this execution was not considered in the final results. Our goal was to collect feedback from the participants and assess the interview plan.

B. CHARACTERIZATION OF PARTICIPANTS

During the analysis phase, 20 candidate subjects were chosen to be interviewed. We focused on practitioners working on agile projects with relevant experience in this topic. Table 1 presents a summary of the participants characteristics.

The selected subjects included 15 Project Managers and 5 Quality Assurance Consultants. Regarding the highest schooling degree, 4 participants have doctoral degree, 6 participants have master degree, 7 participants have bachelor degree and there are 3 participants with associated degree. The distribution for years of professional experience is: 12 participants have between 5 and 10 years of professional experience, 12 participants have between 11 and 20 years and three participants have more than 21 years of professional experience. Regarding the geographic distribution, 16 participants are from Brazil and four from Canada.

C. SURVEY REPORTING

In the last phase, the data collected in the survey were organized, tabulated and analysed. Table 2 presents a summary of the data collected and analyzed in the survey. The table shows the agile smells in relevance order (the most relevant smells are shown first) and the column *Rank* indicates the order in the list. Columns *S1* to *S20* represent the raw data collected in the survey (ie. the answers that each participant provided). These columns are divided in two sides: the left value refers to the answer to Survey-RQ1 and the right value refers to the answer to Survey-RQ2. As explained in the research protocol section, the values vary from 0 to 3 (*No relevant* to *Absolutely relevant*). The *Total* column is the final degree of relevance for the agile smell and was calculated according to the formula in Figure 5.

Note that, as we did not define any tiebreaker criterion, the agile smells *Shared Developers* and *Unplanned Work* are technically tied. The same issue occurs with the agile smells *Large Development Team* and *Long Break Between Iterations*.

D. DATA ANALYSIS DISCUSSION

Most of the agile smells received a positive value for the degree of relevance (ie, they were considered *Slightly relevant*, *Relevant* or *Absolutely relevant*). If we take the top 10 most ranked agile smells in Table 2, they were all considered at least *Slightly relevant* to all the participants.

Figure 6 shows the distribution of the degree of relevance the agile smells received in the survey. The number of *Not relevant* answers was considerably low (only 4.25%, or 34 in 800 responses). The numbers of *Slightly relevant*, *Relevant* and *Absolutely relevant* were, respectively, 32.25% (or 258 in 800), 43.75% (or 350 in 800) and 19.8% (or 158 in 800). These data reveal the identified agile smells are coherent with practices adopted by industry and they could be ultimately used to assess how the agile methods are being applied.

Most participants assigned different degrees of relevance for the presented agile smells. In other words, for most of the participants, some agile smells are more or less relevant than others. That perception was crucial to build the ranking of the most relevant agile smells. Indeed the difference between the relevance of the agile smells at the top and at the bottom of the ranking is significant. While the three top-ranked agile smells vary from 96 to 86 points, the three agile smells at the bottom of the ranking vary from 57 to 49. This difference

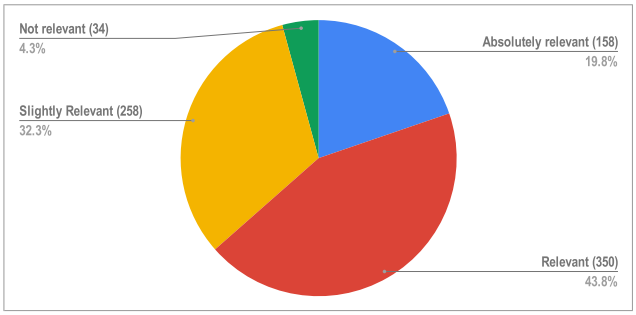


FIGURE 6. Distribution of the degree of relevance according to the survey.

TABLE 3. Agile smell template.

Id	Agile smell's unique identifier
Name	Agile smell's name
Description	A description of the agile smell
Target	The element that the agile smell refers to. It can be: *Organization*, *Project*, *Iteration* or *Team*
Agile Methods	A discussion on how the analyzed agile methods mention the agile smell
Industry Perspective	A discussion of relevant aspects of the agile smell from an industry perspective
Relevance	A percentage value that denotes the degree of relevance of the agile smell according to the survey
Identification Strategy	A description of the identification strategy
Parameters	A description of the parameters that can be used in the identification strategy
References	A list of all references for the agile smell

between the degree of relevance illustrates that the agile smells may impact the adoption of agile methods in different levels.

VI. CONSOLIDATION PHASE

The aim of this phase was to consolidate, complement and organize the agile smells obtained and confirmed in the previous phases as a structured catalogue. Regarding the catalogue structure, the agile smells were described using a template adapted from [75] and shown in Table 3.

The *Id* and *Name* sections indicate, respectively, the unique identifier and the name of the agile smell. The *Description* section presents a brief description of the agile smell and contains: (a) the motivation behind the agile smell and (b) the likely consequences if the agile smell occurs. The *Target* section indicates which element is being assessed when an occurrence of an agile smell is identified. It can assume the values: *Organization*, *Project*, *Iteration* or *Team*. The *Agile Methods* section presents the agile methods practices that motivated the agile smells. Thus, this section establishes a connection between the agile methods analysed during the literature review and the agile smell. The *Industry Perspective* section discusses the agile smell relevance from the perspective of the consulted industry practitioners. The *Relevance* section represents the degree of relevance obtained in the survey converted to the percent of maximum possible score (POMP) [76]. The *Identification Strategy* and *Parameter*

TABLE 4. AS 01: Lower Priority Tasks Executed First.

Id	**AS 01**
Name	**Lower Priority Tasks Executed First**
Description	In an agile project, the development team should focus on higher priority tasks. The *Lower Priority Tasks Executed First* smell is detected when tasks with lower priority are executed before tasks with higher priority. The occurrence of this smell may indicate that the development team has not worked on the highest priority tasks.
Target	Team
Agile Methods	Four agile method explicitly state that higher priority tasks must be executed first: Scrum, Crystal Methods, DSDM and OpenUP. For Scrum, the whole team should focus on the Sprint goal. In Crystal methods, the project leader should prioritize the goals that guide developers to focus on particular areas. In DSDM, to fulfill the principles *Focus on the Business Need* and *Deliver On Time*, DSDM teams must focus on business priorities. OpenUP teams must self-organize around how to accomplish iteration objectives and commit to delivering the results.
Industry Perspective	All survey participants confirmed that working on higher priority tasks is an important agile practice. However, a participant mentioned that exceptions are tolerated in situations where it is not possible to work on high priority tasks. As example, he cited a situation where an available worker does not have the required skills to perform a high-priority task. In this case, the worker is allowed to work on a less important tasks.
Relevance	80%
Identification Strategy	The occurrence of this smell could be detected by assessing the tasks execution history.
Parameters	No parameter was identified for this identification strategy.
References	[16], [21], [9], [14], [15], [20], [22], [19], [44], [23], [45], [17], [10], [46], [48], [55], [49], [51], [69], [8], [52], [53], [60].

sections describe strategies to detect the occurrence of the agile smell in real projects. These sections are designed to support approaches aimed at automatically detecting the occurrence of agile smells in real projects.

For brevity, we have selected the 10 highest ranked agile smells to present in this paper: (Tables 4 to 13).

VII. CATALOGUE USE GUIDELINE

In this section, we describe an use guideline for the catalogue.

Who can use the catalogue? The catalogue can be used by researchers and practitioners that aim at executing an agility assessment and the users may include developers, project managers or quality assurance consultants.

How to use the catalogue? The guideline is based on the method proposed by Moha *et al.* [32] and is composed of four steps as depicted in Figure 7:

- *Step 1: Selection*: In this phase, the agile smells that will be used in the next phases are selected. The selection of an agile smell should be based on the relation between the smell and the agile method adopted by the project. Agile smells that are not related to any agile practice adopted by the project should not be selected in this phase.
- *Step 2: Identification*: This phase aims at identifying the occurrence of the agile smells selected in the previous phase. In the remainder of this guideline, an occurrence of an agile smell will be denoted as an *issue*. The *Identification Strategies* (presented in the catalogue) are manually applied and the project data assessed to identify the issues.

TABLE 5. AS 02: Absence of Frequent Deliveries.

Id	AS 02
Name	**Absence of Frequent Deliveries**
Description	The practice of delivering products continuously and frequently is very important to agile methods and that is almost a mantra among agile software developers. The *Absence of Frequent Deliveries* smell is detected when the development team does not deliver a new version of the software frequently. The occurrence of this smell may indicate that this practice has been jeopardized.
Target	Project
Agile Methods	All analyzed agile methods state that the software project deliveries should be frequent. XP proposes breaking the work in *Small* and *Short Releases* in order to guarantee regular and frequent deliveries. In Scrum, the work is broken in *sprints* which are timeboxed periods (approximately 15 to 30 days) where the development team produces a new executable version of the software. In Crystal methods, the development result product is also incremental and the deliveries interval depends on the length of the project: In Crystal Clear, the delivery intervals are periods of two to three months and in Crystal Orange, the increments can be extended to four months. In FDD, the iterations should take from a few days to a maximum of two weeks. DSDM's philosophy states *"best business value emerges when projects are aligned to clear business goals, deliver frequently and involve the collaboration of motivated and empowered people"*. In ADS, the project is broken into units called Adaptive Development Cycles that typically last between four and eight weeks. OpenUP suggests breaking the work into iterations that take a few weeks.
Industry Perspective	All participants indicated - with different levels of relevance - that the detection of this smell is relevant for an Agility Assessment. No additional comment was given.
Relevance	75%
Identification Strategy	The occurrence of this smell could be detected by assessing the interval between two consecutive Iterations with deliverable.
Parameters	A *Desirable Delivery Interval* parameter could be used to specify the expected duration of the deliveries interval.
References	[16], [21], [9], [14], [15], [20], [22], [19], [44], [23], [45], [17], [10], [46], [47], [40], [48], [49], [50], [51], [18], [8], [52], [53].

TABLE 6. AS 03: Iteration Without a Deliverable.

Id	AS 03
Name	**Iteration Without a Deliverable**
Description	The practice of delivering products continuously and frequently is very important to agile methods and can be considered a mantra among agile software developers. The agile methods state the development team should deliver a new version of the software at the end of each iteration. The *Iteration Without a Deliverable* smell is detected when an iteration does not have an associated deliverable product. The presence of this smell may indicate that the continuous and frequent delivery practice has been jeopardized.
Target	Iteration
Agile Methods	The smell is mentioned by five agile methods: Scrum, FDD, DSDM, ADS and OpenUp. For Scrum, it is desired that the team deliver a new version of the software at the end of each Sprint. In FDD, at least one new feature should be delivered at the end of an Iteration. A key factor for the success of the principle *Deliver on Time* in DSDM is that at the end of each iteration, the team shows a deliverable. In ADS, at least one new component should be delivered at the end of a Development Cycle. The practice *Iterative Development* in OpenUp defines that an iteration should not be extended without any software to be demonstrated.
Industry Perspective	All participants indicated - with different levels of relevance - that the smell is relevant. No additional comment was given.
Relevance	71.67%
Identification Strategy	A strategy to assess whether an iteration has a deliverable is to use a specific field to describe the deliverable of an iteration. With such field, the occurrence of this smell could be detected by assessing this field.
Parameters	A *Tolerated Number of Consecutive Iterations Without Deliverable* parameter could be used to specify a tolerable number of consecutive iterations without a deliverable.
References	[16], [21], [9], [14], [15], [20], [22], [19], [44], [23], [45], [17], [10], [46], [40], [48], [49], [51], [18], [8], [53].

- *Step 3: Validation*: In this phase, the issues detected in the previous phase are analyzed and the false-positives are discarded.

TABLE 7. AS 04: Goals Not Defined.

Id	AS 04
Name	**Goals Not Defined**
Description	Agile development teams need to know exactly what they are working on and the goals of the project and iterations should be clear and well-defined. The *Goals Not Defined* smell is detected when the goals of the project or of a given iteration are not defined. The presence of this smell may indicate the development team does not have a clear view of the goals and therefore could not choose the most important work to do.
Target	Project/Iteration
Agile Methods	*Goals should be clear and well-defined* is a practice mentioned by the following methods: Scrum, Crystal, DSDM, ADS and OpenUp. Scrum states that the iterations (called sprints in Scrum) should have well-defined goals. In Crystal methods, goals should be clear and developers should know exactly what the goals of the project are. The principle *Focus On The Business Need* in DSDM defines that every decision taken during a project must be guided by the project goals. Thus, it is important that these goals are well-defined and communicated to all team. In ADS, the development process should be mission-oriented (or goal-oriented) and the activities in each iteration (called cycle in ADS) must be aligned with the project mission. Thus, having a well-defined and clear goal is a key factor for ADS method. The development team in OpenUP method should be self-organized around how to accomplish iteration objectives.
Industry Perspective	All participants indicated - with different levels - that the agile smell is relevant. No additional comment was given.
Relevance	68.33%
Identification Strategy	Decide what is "clear and well-defined" could not be an easy decision specially since there is no pre-defined format to specify "goals" in agile method. Thus, we propose a simpler strategy that only verifies if the goals of the project and iterations are defined. To achieve this verification, specific fields should be used to describe the goals.
Parameters	A *Min Length* parameter could be used to specify the minimum length the goal description should have.
References	[16], [21], [20], [22], [19], [38], [23], [17], [10], [46], [48], [51], [69], [8]. [52].

- *Step 4: Reporting*: The positive issues are consolidated and a report is generated.

VIII. RELATED WORK

Studies that aim at consolidating the body of knowledge around agility assessment are in some level related to this paper. We divide those studies in two categories: (a) studies that aim at identifying agile practices and characteristics to support the adoption of agile methods, and (b) studies that aim at identifying requirements to support agility assessment.

A. MAPPING AGILE PRACTICES

This category includes studies that identified agile practices or characteristics to support the adoption of agile development. We consider these studies related to our research because the identified agile practices could be ultimately used as input to an agility assessment process.

Miller [38] conducted one of the first studies that aimed at investigating characteristics of agile development. Among the identified characteristics, we can mention: *modularity on development process*, *iterative with short cycles*, *time-bound with iteration cycles from one to six weeks*, *parsimony in development processes removing unnecessary activities*, *adaptive with possible emergent new risks*, *incremental process approaches that allow functioning application building in small steps*, *convergent (and incremental) approach*, and *people-oriented*.

TABLE 8. AS 05: Iteration Without an Iteration Planning.

Id	AS 05
Name	Iteration Without an Iteration Planning
Description	Iteration planning is an important success factor in agile methods. Normally an iteration plan is elaborated with the main stakeholders (developers and customer) that together decide what should be developed in the iteration. The *Iteration Without an Iteration Planning* smell is detected when there is no planning associated with a given iteration. The presence of this smell may indicate that the iterations are not being planned properly.
Target	Iteration
Agile Methods	Iteration planning is mentioned by all agile methods investigated. The *Planning Game* practice in XP promotes a close interaction between developers and customers. Developers estimate the effort for implementing customer stories and customers then decide about timing and scope of the deliverables. Scrum proposes the *Sprint Planning* which is a meeting divided in two parts: in the first part, all stakeholders (customer, scrum master and developers) select from product backlog the items that should be worked in the iteration. In the second part, the team discusses technical issues related to selected items and decides what features could be delivered in the iteration. Crystal methods define a planning meeting to decide the next increment of the system. In FDD, development is feature-oriented which includes the creation of a high-level plan where features are organized according to their priority. Before each iteration, customer and team decide together which features should be developed in the next iteration. In DSDM, the scope of an iteration is planned beforehand. Planning the cycles in ADS is part of the iterative process. ADS method also proposes *Joint Application Development* (JAD) sessions which are workshops where developers and customer representative discuss product features and decide the components that will be included in the next cycle. The *Iteration Plan* practice in OpenUp suggests planning an iteration in detail only when it is due to start. An iteration planning meeting should be held by the whole project team who decide the iteration scope.
Industry Perspective	All the participants considered the Iteration Planning relevant. One participant mentioned that in some Projects the Iteration Planning is divided in two parts (different from Scrum division mentioned above). In the first part (which the participant called Pre-Iteration Planning) a team member representative (usually the most experienced) and a business analyst discuss details related to items that are candidates to be developed in the next iteration. The goal of the first part is to anticipate potential technical problems (inconsistent business rules, incomplete or hard-to-implement requirements, business processes not mapped, etc). In case any problem is detected, the issue should be resolved before the Iteration Planning.
Relevance	67.5%
Identification Strategy	We propose three strategies to verify the presence of the *Iteration Without an Iteration Planning* smell. The first strategy is to check if there is a task in the iteration plan that represents the *Iteration Planning* meeting. Another strategy is to verify if there is a task in the iteration plan that produces an *Iteration Plan* artifact. A third strategy is to check if all the tasks in the iteration plan are estimated before starting the iteration.
Parameters	No parameter was identified for the evaluation strategy.
References	[16], [21], [9], [14], [15], [20], [22], [19], [44], [23], [45], [17], [10], [46], [70], [66], [67], [47], [40], [71], [54], [48], [55], [64], [49], [50], [18], [8], [56], [57], [58], [59], [52], [60], [61].

In [77], Sidky proposed a structured method to guide organizations in adopting agile development. The method is divided in 4 stages: *Discontinuing Factors*, *Project-level Assessment*, *Organizational Readiness Assessment*, and *Reconciliation*. In each stage, different actions are proposed to cover the following practices: *Delivering Customer Value by Embracing Change*, *Planning to Deliver Software Frequently*, *Human-centric*, *Technical Excellence*, and *Customer Collaboration*. This method was one of the first structured approaches to guide the use of agile practices. However, it differs from our study mainly by specifying the practices in a generic way and by not giving an indication of how to check if they were properly adopted.

TABLE 9. AS 06: Complex Tasks.

Id	AS 06
Name	Complex Tasks
Description	Complex tasks should be avoided in agile projects. They should be decomposed by the development team into simpler tasks. The *Complex Tasks* smell is detected when there are complex tasks in a given iteration. The presence of this smell may indicate that the developers are not properly breaking complex tasks into simpler tasks.
Target	Iteration
Agile Methods	The motivation for avoiding complex tasks in Scrum is derived from a technique for project management called a *Burndown Chart*. A *Burndown Chart* reflects the daily progress of the team and decreases according to the number of finished tasks. The chart is expected to decrease daily after the *Daily Meeting*. Otherwise, a delay in working-in-progress is detected. The problem with complex tasks - those whose duration exceeds 8 hours - is that they may give a false indication that the work-in-progress is not evolving. Therefore, simple tasks make the iteration management easier and more reliable since it is expected that each developer finishes at least one task per day. Avoiding complex tasks is also explicitly mentioned by FDD that suggests that the features should be small enough to be implemented in a few hours or days.
Industry Perspective	All participants indicated - with different levels - that the agile smell is relevant. No additional comment was given.
Relevance	65%
Identification Strategy	A strategy to identify the presence of the *Complex Tasks* smell is to verify whether the tasks estimates exceed an allowable threshold.
Parameters	A *Maximum Estimation Allowed* parameter could be used to configure the threshold that used to identify complex tasks.
References	[16], [17], [46], [62], [47], [40], [55], [49], [8], [52], [61].

TABLE 10. AS 07: Iteration Without an Iteration Retrospective.

Id	AS 07
Name	Iteration Without an Iteration Retrospective
Description	Retrospective meetings represent opportunities for the development team to reflect on how they are working and improve the method when necessary. The *Iteration Without an Iteration Retrospective* smell is detected when there is no retrospective meeting associated with a given iteration. The presence of this smell may indicate that an important opportunity for improvement prescribed by agile methods is being wasted.
Target	Iteration
Agile Methods	Retrospective meetings are mentioned by three agile methods: Scrum, Crystal methods and ADS. Scrum defines *Sprint Retrospect* as a meeting that usually follows the *Sprint Review*, where the development team (and only it) gives and receives feedback on the process followed during the Sprint. Crystal encourages a practice called *Reflective Improvement* in which developers take a break from regular development and try to improve their processes. The *Learning Loop* principle in ADS is also based on retrospective meetings that are usually performed after each cycle.
Industry Perspective	All participants indicated - with different levels - that the agile smell is relevant. No additional comment was given.
Relevance	64.17%
Identification Strategy	Two strategies were proposed to verify the presence of a Retrospective Meeting. The first strategy consists in checking if there is any task associated with the iteration plan that represents the Retrospective meeting. The second strategy is to verify if there is any task in the iteration plan that produces an *Iteration Retrospective Minute* artifact.
Parameters	A *Tolerated Number of Consecutive Iterations Without Retrospective* parameter could be used to specify a tolerable number of consecutive iterations without retrospective meetings.
References	[16], [21], [20], [22], [19], [23], [17], [10], [46], [47], [54], [48] [55], [49], [50], [51], [18], [56], [59], [57], [58], [52], [53], [60], [61].

In [78], Shore and Warden described a set of practices to guide the adoption of XP and other agile methods. The practices are divided into two categories: *Practicing XP* and *Mastering Agile*. The first category is focused on the XP

TABLE 11. AS 08: Absence of Timeboxed Iteration.

Id	AS 08
Name	**Absence of Timeboxed Iteration**
Description	The *Timeboxed Iteration* practice defines that all iterations should have a fixed time duration. Thus, an iteration should not be extended or shortened to fit planned or unplanned features. The *Absence of Timeboxed Iteration* smell is detected when an iteration is shorter or longer than the predefined duration. The presence of this smell may indicate the timeboxed iteration practice has not been applied properly.
Target	Iteration
Agile Methods	The *Timeboxed Iteration* practice is mentioned by four methods: Scrum, DSDM, ADS and OpenUp. Scrum method defines that an iteration (called sprint in Scrum) should be timeboxed. In DSDM, the *Deliver On Time* principle states that delivering a solution on time is a very desirable outcome for a project and is quite often the single most important success factor. In order to achieve this principle, DSDM teams should need to timebox work. ADS method argues that ambiguity in complex software development can be alleviated by fixing tangible deadlines on a regular basis. The *Iterative Development* practice in OpenUp defines that do not extend an iteration in order to finish work.
Industry Perspective	Regarding the survey, there was no unanimity among the participants. Some participants said timeboxing should be rigidly followed. Other subjects said that timeboxing is desired, but not applied as a rigid rule. Changes in iteration duration are indeed a common practice. For these participants, it is preferable to extend an iteration in order to include important features than achieve timeboxing with less features.
Relevance	63.33%
Identification Strategy	Timeboxing could be verified assessing if there was any variation in the iteration duration after it has been planned.
Parameters	A *Tolerance Of Change* parameter (absolute or percentage value) could be used to indicate the maximum tolerated variation. For example, a 5% tolerance means that an iteration could have their duration shortened or extended by 5% of its baseline duration.
References	[16], [22], [38], [17], [10], [46], [54], [48], [55], [49], [50], [18], [8], [53].

TABLE 12. AS 09: Iteration Started without an Estimated Effort.

Id	AS 09
Name	**Iteration Started without an Estimated Effort**
Description	The scope and duration of the iterations in an agile project are typically defined by the development team that must commit to the iteration goals and deadlines. The *Iteration Started without an Estimated Effort* smell is detected when an iteration that contains non-estimated tasks is started. The presence of this smell may indicate that the development team is committed to a deadline without a good understanding of the effort to deliver the iteration scope.
Target	Iteration
Agile Methods	Understanding the effort necessary to developer all the features selected to a given iteration is a key success factor for all agile methods investigated. In the *Planning Game* practice proposed in XP, the developers should estimate the effort for implementing customer stories and customers then decide about timing and scope of the deliverables. Scrum proposes the *Sprint Planning* meeting where the developers discuss and scrutinize technical issues related to backlog to decide what features they are able to deliver after the next iteration. Similarly, Crystal methods define a planning meeting where the team decides the next increment of the system. In FDD, development is feature-oriented which includes the creation of a high-level plan where features are organized according to their priority. Before each iteration, customer and team decide together, based on the priority of the items and the team productivity, which features should be developed in the next iteration. In DSDM, the scope of an iteration is planned beforehand. Planning the cycles in ADS is part of the iterative process. ADS method also proposes *Joint Application Development* (JAD) sessions which are workshops where developers and customer representative discuss product features and decide the components that will be included in the next cycle. The *Iteration Plan* practice in OpenUp suggests planning an iteration in detail only when it is due to start. An iteration planning meeting should be held by the whole project team who decide the iteration scope.
Industry Perspective	All participants indicated - with different levels - that the agile smell is relevant. No additional comment was given.
Relevance	62.5%
Identification Strategy	A strategy to verify the occurrence of this agile smell is to check whether all the tasks selected to a given open iteration are estimated (have an effort estimated).
Parameters	No parameter was identified for the evaluation strategy.
References	[16], [21], [20], [22], [19], [23], [17], [10], [46], [47], [71], [54], [48], [55], [49], [18], [8], [56], [58], [59], [52], [53], [60].

method and contains 37 practices such as *Pair Programming*, *Retrospectives*, *Trust*, *Sit Together*, *Real Customer Involvement*, *Stand-Up Meetings*, *Version Control*, *Release Planning*, and *Incremental Requirements*. The second part contains generic agile practices such as *Tune and Adapt*, *Work in Small, Reversible Steps*, *Deliver Frequently*, and *Deliver Business Results*. The practices represent a useful asset for guiding the adoption of agile practices, especially XP, into the software industry.

Abrantes and Travassos [40] conducted a study to identify the most commonly used agile practices. The authors identified a set of 12 practices: *test driven development*, *continuous integration*, *pair programming*, *planning game*, *onsite customer*, *collective code ownership*, *small releases*, *metaphor*, *refactoring*, *sustainable pace*, *simple design*, and *coding standards*.

Although many others papers aim at mapping agile practices [46], [67], [79]–[82], these studies do not investigate agile methods from the perspective of this study, namely, trying to identify practices that impair the use of agile development and indicate how these practices can be checked in real scenarios. The agile practices presented in the studies above are typically described for educational purposes rather than focusing on agility assessment. That is, the intention of the authors is to guide development teams and organizations in how to apply the agile practices. There is little focus on verifying whether the agile practices have been properly adopted.

B. AGILITY ASSESSMENT REQUIREMENTS

Studies in this category describe practices and requirements to be checked in an agility assessment.

Yatzeck proposed in [63] a two-checklist method to aid the adoption and assessment of the agile process in large companies. The first checklist is focused on guiding the adoption of Scrum and it is composed of 10 items. The second checklist, called "*You Should Immediately Be Suspicious If*", describes 8 practices that may indicate misuse of agile practices: 1) *"There is no high-level architecture"*, 2) *"There is no plan"*, 3) *"There is no project dashboard, or you don't have access"*, 4) *"You aren't invited to an iteration planning meeting and a showcase for every iteration"*, 5) *"You don't get any escalations coming out of the planning workshop"*, 6) *"The team performs perfectly in Iteration 1"*, 7) *"You aren't welcome to join daily standup Scrum meetings as an observer"*, and 8) *"You can't get metrics about software quality"*. These items are similar to the Bad Agile Smells proposed in this study since they describe practices that

TABLE 13. AS 10: Iteration Without an Iteration Review.

Id	AS 10
Name	**Iteration Without an Iteration Review**
Description	The iteration review is a meeting where the development team presents to the product owner what was accomplished during the previous iteration. Typically, there is a software demonstration showing the new features and a discussion of what is being delivered. The *Iteration Without an Iteration Review* smell is detected when there is no review associated with a given iteration. The presence of this smell may indicate the development team is missing an important opportunity to present the results of the iteration to the product owner.
Target	Iteration
Agile Methods	Review meetings are explicitly mentioned by three agile methods: Scrum, Crystal methods and ADS. Scrum defines *Sprint Review* as a meeting that should be held on the last day of the iteration to the team presents the results (ie the features added to the software) to the stakeholders. The participants assess the new features and may decide for adjustments or even new features that change the direction of the software development. Similarly, Crystal methods propose a review meeting just after the end of each iteration to the team presents the resulting software. The *Learning Loop* principle in ADS also contains a review meeting that should be performed after each cycle. This meeting should be performed in the presence of a customer representative (called customer focus-group).
Industry Perspective	All participants indicated - with different levels - that the agile smell is relevant. No additional comment was given.
Relevance	61.67%
Identification Strategy	Two strategies were proposed to verify the presence of a Review Meeting. The first strategy consists in checking if there is any task associated with the iteration plan that represents the Review meeting. The second strategy is to verify if there is any task in the iteration plan that produces an *Iteration Review Minute* artifact.
Parameters	A *Tolerated Number of Consecutive Iterations Without Review* parameter could be used to specify a tolerable number of consecutive iterations without retrospective meetings.
References	[16], [21], [20], [22], [19], [23], [17], [10], [46], [47], [71], [54], [48], [55], [49], [18], [8], [56], [58], [59], [52], [53], [60].

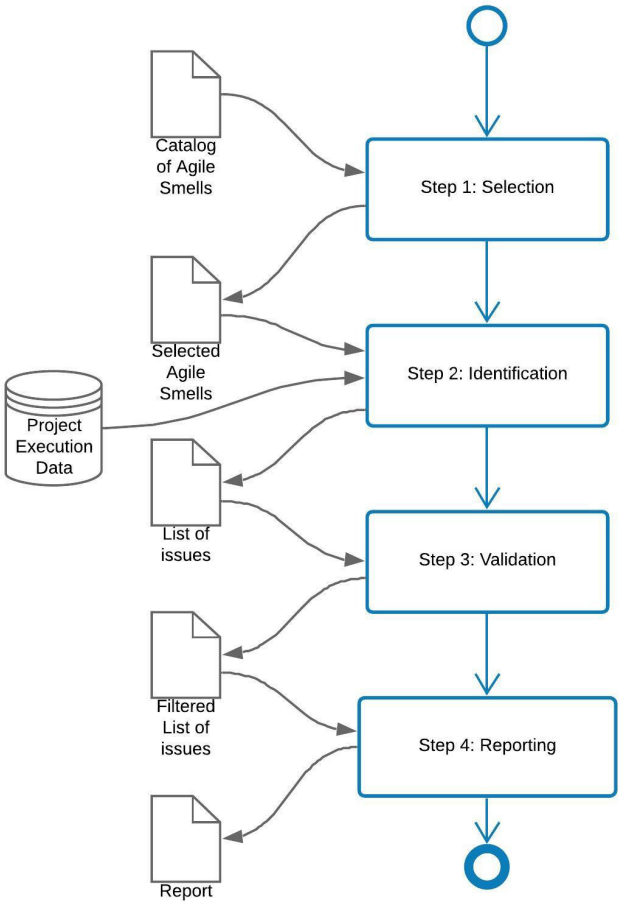


FIGURE 7. Catalogue use guideline.

may jeopardize the adoption of agile methods. However, the study differs from ours in three aspects: (a) the practices are described in a generic way and there is no indication of how they could be checked in real scenarios. Therefore, the verification of these practices may be threatened by the bias of the person performing the agility assessment that has to interpret the practice and determine how to check it; (b) the items are focused on the Scrum method; and (c) there is no clear relation between the items and the agile practices that motivated them. The author did not explain the origin of the items. The author also failed in not describing the criteria that define the target companies (large companies).

Hermida proposed an online agility assessment approach called "Abetterteam" [83]. The tool has a questionnaire composed of 30 three-option questions. The author claimed the tool is able to verify the adoption of the practices proposed by Shore and Warden in [78]. However, the author did not indicate how the questionnaire is related to the practices proposed by Shore and the rationale behind the assessment result.

Krebs *et al.* proposed an agility assessment model called *Agile Journey Index (AJI)* [84] that aids organizations in improving their application of the agile method. The model looks at 19 key practices and divides them into 3 categories: *Plan*, *Do* and *Feedback*. The assessment consists of rating each practice on a scale of 1 to 10. Although the model specifies criteria for each score, the evaluation of these criteria depends on qualitative analysis and there is no indication of how to identify the occurrence of these practices in real projects. Another drawback of this model is that it considers only Scrum practices and neglects other agile methods.

In [85], Williams *et al.* proposed the *Comparative Agility*^{TM} (CA) method to aid organizations in determining their relative agile capability compared to other companies who responded to CA. The tool, that is available as a survey-tool, assesses agility using seven dimensions: *Teamwork*, *Requirements*, *Planning*, *Technical Practices*, *Quality*, *Culture*, and *Knowledge Creation*. Each dimension has between three and six characteristics (32 in total) and each characteristic is made up of approximately four agile practices (125 in total). For each practice, the respondent indicates the truth of the practice using a five-point Likert scale: *True*; *More true than false*; *Neither true nor false*; *More false than true*; or *False*. Although the approach uses an innovative assessment technique (by comparing the answers given by the company with a global trend), the authors neglected to indicate how the practices were identified, how they are related to the agile methods, and how they can be verified. One of the questions that composes the method, for example, is *"Team members leave planning meetings knowing what needs to be done and*

have confidence they can meet their commitments". There are no clear criteria to check the occurrence of this practice.

The *Enterprise and Team Level Agility Maturity Matrix* [86] is an agility assessment method available as a spreadsheet divided into two sections: one for describing the Organization and another for describing the Development Team. There are a number of agile indicators for each section (14 organizational indicators and 37 team indicators) and each indicator ranges from a '0' (impeded) to a '4' (ideal). For each cell in the matrix, there is a simple explanation of what it means to be at that level for that indicator.

IBM DevOps Practices Self-Assessment [87] is another agility assessment approach available as a web application. The solution contains 15 questions divided into 4 areas: *Demographic*, *Practices*, *Strategies*, and *Motivation*. The authors claimed the tool can ''evaluate the state of an organization's software delivery approach''. However, there are no indications of how the questions were formed, how the answers should be analyzed and how the results are related to agile practices.

The *Scrum Checklist* [88] is a tool to help development teams getting started with Scrum, or assessing their current implementation of Scrum. The checklist is made up of 80 items divided into 4 groups: *The Bottom Line*; *Core Scrum*; *Recommended But Not Always Necessary*; *Scaling*; and *Positive Indicators*. According to the author, the items on the checklist are not rules and therefore were not designed to be verifiable or to produce a measure that indicates the level of compliance with Scrum. Instead, they are guidelines that might be used by the team as a discussion tool at the retrospective meetings. Examples of items on the checklist are *"Whole team believes plan is achievable?"* or *"Having fun? High energy level?"*.

There are also models that aim to assess team members individually. In [89], Campbell and MacIver define a self-assessment model named *Agility Maturity Self-Assessment* that intends to identify the skills of individuals in six areas: *Agile Teams*, *Agile Leadership*, *Agile Project Management*, *Agile Communication/Promotion*, *Business Value*, and *Risk Management*. The questions have the following structure *"How experienced are you in the given area…"*. The author did not provide any indication of how to analyze the answers. In [90], Ribeiro proposed a survey where the participants can assess their skill in agile development by answering a questionnaire composed of 25 questions (including an open question). The author did not provide indications of how to analyze the answers and to assess the skill of the individuals. Other self-assessment tools and methods are proposed in [91], [92] and [93]. In [94], an extensive case study was conducted to evaluate 22 agile maturity self-assessment surveys according to seven criteria namely: a) comprehensiveness, b) fitness for purpose, c) ability to discriminate, d) objectivity, e) conciseness, f) generalizability, and g) suitability for multiple assessment. The authors concluded none of the evaluated approaches fully satisfy all of the criteria but are helpful to some degree.

Although these studies may be useful for guiding the assessment of developers in specific skills related to agile development, they differ from this study in not considering the assessment of projects and organizations and not providing objective indications of how the practices can be checked in an agility assessment.

Other studies proposed methods, tools or requirements to support agility assessment [95]–[99], and [100]. However, these proposals differ from the catalogue of agile smells in three ways namely: (a) the practices are usually described in a vague way which makes their verification jeopardized by the bias of the professional performing the assessment. (b) there is no indication of how to verify the occurrence of these practices; and (c) there is no clear relationship between the proposed criteria for agility assessment and any agile method.

A comprehensive study conducted by Chronis [101] investigated three questionnaire based approaches to measure the agility of development teams: *Team Agility Assessment* (TAA) [102], *Perceptive Agile Measurement* (PAM) [103], and *Objectives Principles Strategies* (OPS) [104]. These approaches do not focus on a specific agile methodology. Instead, they try to evaluate the degree of agility based on generic agile principles and values. The approach proposed by Leffingwell [102] (TAA), for example, assesses the agility of a development team by considering six aspects: a) Product Ownership, b) Release Planning and Tracking, c) Iteration Planning and Tracking, d) Team, e) Testing Practices, and f) Development Practices/Infrastructure. On the other hand, the method proposed by So and Scholl [103] (PAM) focuses on eight agile areas (that the authors named scales): a) Iteration Planning, b) Iterative Development, c) Continuous Integration and Testing, d) Stand-Up Meetings, e) Customer Access, f) Customer Acceptance Tests, g) Retrospectives, and h) Collocation. The framework proposed by Soundararajan (OPS) focuses on 17 agile strategies namely: a) Iterative progression, b) Incremental development, c) Short delivery cycles, d) Evolutionary requirements, e) Continuous feedback, f) Refactoring, g) Test-first development, h) Self-managing teams, i) Continuous integration, j) Constant velocity, k) Minimal documentation, l) High bandwidth communication, m) Retrospection, n) Client-driven iterations, o) Distribution of expertise, p) Configuration management, and q) Adherence to standards.

Other studies aimed at assessing the agility of a software development methodology. Nebe and Baloni [105] investigated the integration of agile development and User-Centred Design (UCD) and proposed a checklist to assess the user-centeredness of agile processes.

Qumer and Henderson-Sellers [106] introduce a framework to assist the selection and assessment of agile development methods. The framework is composed of seven components: 1) Knowledge-Base, 2) Process Fragment and Process Composer, 3) Publisher, 4) Registry, 5) Agility Calculator, 6) Knowledge-Transformer, and 7) Visualizer. The Agility Calculator, for example, generates a report and may assist organizations in making decisions about

the selection or adoption of an agile method or method fragments.

It is important to note that most of the agile smells proposed in this study are intimately related to some agile practices described in the studies presented previously. For example, some of the smells presented in Section VI are related to the *Timeboxed* practice (mentioned in several studies already discussed). This relationship is expected as almost all the agile practices derived from the same set of values and principles proposed in the Manifesto for Agile Development [7] and shown in Section VIII. However, we consider our study an important contribution because the proposed catalogue is focused on providing elements (agile smells) that can be directly checked in an agility assessment. Thus, for each smell, the catalogue indicates the motivating agile practice and at least one strategy to identify its occurrence in a real software project.

IX. RESULTS AND THREATS TO VALIDITY

In this section we present the results of this study and discuss some threats to validity.

A. RESULTS

The main contribution of this study is the catalogue of agile smells shown in Tables 4 to 13. These agile smells were organized to aid managers and developers assessing the agility of their projects, iterations and agile teams. This catalogue of agile smells fills a gap in the agile literature by making explicit an important set of agility assessment requirements. While the *Manifesto for Agile Development* [7] and other agile methods [8]–[10], [20]–[23] present the agile values, principles and practices in a rather vague description [41], [42], this catalogue presents objective criteria to assess whether some agile practices have been properly adopted. We present the agile smells in a uniform vocabulary using templates that were designed to support practitioners in agility assessment. The *Identification Strategy* section provides a guideline to detect the occurrence of these agile smells in real projects.

The agile smells can be used to assess agility in the level of organization, project, iteration or team. Agile smells such as *Goals Not Defined* and *Dependence on Internal Specialists* are designed to assess the project. Occurrences of these agile smells indicate adjustments in the project may be necessary. The agile smell *Dependence on Internal Specialists* may reveal a failure in the project planning.

Agile smells such as *Iteration Without a Deliverable*, *Iteration Without an Iteration Retrospective*, *Absence of Timeboxed Iteration* and *Iteration Without an Iteration Planning* are designed to assess the iterations. Occurrences of these agile smells indicate adjustments in the iteration planning and iteration execution may be necessary. The agile smells *Iteration Without an Iteration Retrospective* and *Iteration Without an Iteration Planning* indicate the lack of important meetings prescribed by the agile methods.

The *Lower Priority Tasks Executed First* agile smell assesses the agile team. The occurrence of this agile smell may indicate the agile team is not working on the most important tasks as prescribed by almost all agile methods.

The *Shared Developers* agile smell targets the organization. An occurrence of this smell may indicate the organization is not properly allocating developers based on the agile values.

B. THREATS TO VALIDITY

This section discusses the threats to the validity of this research and the actions that were taken to avoid them.

External Validity. This refers to the degree to which the identified agile smells are relevant to the industry. To confirm the lack of bias of the extraction method used in the literature review and to confirm the relevance of the identified agile smells to the industry, a survey with experienced practitioners from two different countries was conducted.

Construct Validity. This validates whether the research explores what it claims to be exploring. A threat in this category is not reaching the ''state of the art'' about agile development. As a significant part of the body of knowledge about Agile Methods is created by software engineering (SE) practitioners that usually do not publish in academic forums [107], we decided to include in the literature review the grey literature (non-peer-reviewed material).

Internal Validity. This validates whether the agile smells identified in the literature review are internally valid. A risk to this validity came from the fact there is no use of the term ''smell'' in the current literature. We thus sought to mitigate this threat by defining objective criteria to extract the agile smells from the selected papers.

Conclusion Validity. This threat is related to problems that can impact the reliability of our conclusions. A risk in this category regards the survey sampling size. The survey was conducted with a sampling that is not representative enough to allow us to affirm that the set of identified agile smells represents the most relevant. So, there may be some variation in the ranked list whether we conduct a survey with a more representative sampling.

X. CONCLUSION

The goal of this study was twofold: (a) identify a set of practices that may impair the proper adoption of agile development (AKA agile smells); and (b) propose strategies to identify the occurrence of such practices in real projects.

The study was organized into three phases: Elicitation, Confirmation, and Consolidation. In the *Elicitation* phase, we conducted a literature review including peer-reviewed academic publications and ''grey'' literature that extracted an initial set of 20 agile smells. In the *Confirmation* phase, this set of agile smells was the subject of a survey that aimed at characterizing the smells according to their relevance. Finally, in the *Consolidation* phase, the agile smells were consolidated and organized as a catalogue.

The catalogue aims at helping practitioners conduct agility assessment by (a) describing a set of practices that may impair the proper use of agile methods and (b) presenting strategies to identify the occurrence of such practices in real projects.

The catalogue can be used by researchers and industry practitioners who intend to execute agility assessment of their projects or organizations. The catalogue can also be used in research that aims at automatically detecting agile smells. As proposed by methods such as DECOR [32], making the smells explicit through a structured catalogue is a prelude to an automatic approach toward detection.

In future work, we will (a) evaluate the practical use of the catalogue guideline with case studies in the context of industry; (b) investigate how to measure eventual ''technical debts'' caused by these smells; (c) extend the catalogue including new agile smells; (d) identify the most relevant smells through a more comprehensive survey with a more representative sampling; and (e) investigate the relationship between the agile smells and specific agile methods.

APPENDIX

LITERATURE REVIEW SELECTED STUDIES

The 55 studies selected for full consideration in the systematic literature review are listed below.

P1: Schwaber, K.: SCRUM development process. In: Business Object Design and Implementation, pp. 117–134. Springer London (1997) [16].

P2: Stapleton, J.: Dynamic systems development method: the method in practice. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1997) [21].

P3: Beck, K.: Embracing change with extreme programming. Computer **32**(10), 70–77 (1999) [14].

P4: Beck, K.: Extreme programming explained: embrace change. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2000) [14].

P5: Cunningham, W.: Extreme programming. http://www.-extremeprogramming.org/ (1999), accessed: 2017-12-01 [9].

P6: Luca, J.D.: Feature driven development FDD. http://www.-featuredrivendevelopment.com/ (1999), accessed: 2017-12-01 [20].

P7: Highsmith, III, J.A.: Adaptive software development: a collaborative approach to managing complex systems. Dorset House Publishing Co., Inc., New York, NY, USA (2000) [22].

P8: Miller, G.G.: The characteristics of agile software processes. In: Proceedings of the 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS39). TOOLS '01, IEEE Computer Society, Washington, DC, USA (2001) [38].

P9: Palmer, S.R., Felsing, M.: A practical guide to feature-driven development. Pearson Education, 1st edn. (2001) [].

P10: Maurer, F., Martel, S.: Extreme programming: rapid development for web-based applications. IEEE Internet Computing **6**(1), 86–90 (2002) [44].

P11: Newkirk, J.: Introduction to agile processes and extreme programming. In: Proceedings of the 24th International Conference on Software Engineering. ICSE 2002. pp. 695–696 (May 2002) [45].

P12: Lindvall, M., Basili, V.R., Boehm, B.W., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams, L.A., Zelkowitz, M.V.: Empirical findings in agile methods. In: Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002. Springer-Verlag, London, UK, UK (2002) [39].

P13: Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile software development methods - review and analysis. Tech. Rep. 478, VTT Publications (2002) [23].

P14: Schwaber, K., Beedle, M.: Agile software development with scrum. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edn. (2001) [17].

P15: Cockburn, A.: Agile software development. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2002) [10].

P16: Martin, R.C.: Agile software development: principles, patterns, and practices. Prentice Hall PTR, Upper Saddle River, NJ, USA (2003) [46].

P17: Nisar, M., Hameed, T.: Agile methods handling offshore software development issues. In: Proceedings of INMIC 2004 - 8th International Multitopic Conference. pp. 417–422 (2004). [62].

P18: McMahon, P.: Extending agile methods: a distributed project and organizational improvement perspective. CrossTalk (5), 16–19 (2005) [70].

P19: Miller, G.: Agile software development for the entire project. CrossTalk **18**(12), 9–12 (2005) [108].

P20: Pikkarainen, M., Salo, O., Still, J.: Deploying agile practices in organizations: a case study, vol. 3792 LNCS (2005) [109].

P21: Ambler, S.: Survey says: agile works in practice. Dr. Dobb's Journal **31**(9), 62–64 (2006) [110].

P22: Cao, L., Ramesh, B.: Agile software development: ad hoc practices or sound principles? IT professional **9**(2), 41–47 (2007) [65].

P23: Thomas, J.: Introducing agile development practices from the middle. In: Engineering of Computer Based Systems, 2008. ECBS 2008. 15th Annual IEEE International Conference and Workshop on the. pp. 401–407. IEEE (2008) [66].

P24: Kautz, K., Pedersen, C., Monrad, O.: Cultures of agility - agile software development in practice. In: ACIS 2009 Procedings - 20th Australasian Conference on Information Systems. pp. 174–184 (2009) [111].

P25: Batra, D.: Modified agile practices for outsourced software projects. Communications of the ACM **52**(9), 143–148+10 (2009) [112].

P26: Misra, S.C., Kumar, V., Kumar, U.: Identifying some important success factors in adopting agile software development practices. J. Syst. Softw. **82**(11), 1869–1890 (nov 2009) [67].
P27: Li, J.: Research and practice of agile unified process. In: ICSTE 2010 - 2010 2nd International Conference on Software Technology and Engineering, Proceedings. vol. 2, pp. V2340–V2343 (2010) [68].
P28: Williams, L.: Agile software development methodologies and practices. In: Advances in Computers, vol. 80, pp. 1–44. Elsevier (2010) [47].
P29: Abrantes, J.F., Travassos, G.H.: Common agile practices in software processes. In: 2011 International Symposium on Empirical Software Engineering and Measurement. pp. 355–358 (Sept 2011) [40].
P30: Shi, Z., Chen, L., Chen, T.E.: Agile planning and development methods. In: ICCRD2011 - 2011 3rd International Conference on Computer Research and Development. vol. 1, pp. 488–491 (2011) [71].
P31: Poppendieck, M., Cusumano, M.: Lean software development: a tutorial. IEEE Software **29**(5), 26–32 (2012) [113].
P32: Sletholt, M., Hannay, J., Pfahl, D., Langtangen, H.: What do we know about scientific software development's agile practices? Computing in Science and Engineering **14**(2), 24–36 (2012) [54].
P33: Dyck, S., Majchrzak, T.: Identifying common characteristics in fundamental, integrated, and agile software development methodologies. In: Proceedings of the Annual Hawaii International Conference on System Sciences. pp. 5299–5308 (2012) [114].
P34: Alzoabi, Z.: Agile software: body of knowledge. In: Business Intelligence and Agile Methodologies for Knowledge-Based Organizations: Cross-Disciplinary Applications, pp. 14–34. IGI Global (2012) [48].
P35: Yatzeck, E.: A corporate agile 10-point checklist [63].
P36: Meyer, B.: Agile!: the good, the hype and the ugly, Agile!: the good, the hype and the ugly, vol. 9783319051550 (2014) [55].
P37: Papatheocharous, E., Andreou, A.: Empirical evidence and state of practice of software agile teams. Journal of Software: Evolution and Process **26**(9), 855–866 (2014) [2].
P38: Chagas, L.F., de Carvalho, D.D., Lima, A.M., Reis, C.A.L.: Systematic literature review on the characteristics of agile project management in the context of maturity models. In: International Conference on Software Process Improvement and Capability Determination. pp. 177–189. Springer (2014) [64].
P39: Moran, A.: Agile software development. In: Agile Risk Management, pp. 1–16. Springer (2014) [49].
P40: Diebold, P., Dahlem, M.: Agile practices in practice: a mapping study. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. p. 30. ACM (2014) [50].
P41: Berteig, M.: Rules of scrum. http://www.agileadvice.com/rules-of-scrum/(2015), accessed: 2017-12-01 [18].
P42: Ghani, I., Bello, M.: Agile adoption in IT organizations. KSII Transactions on Internet and Information Systems **9**(8), 3231–3248 (2015) [51].
P43: Gregory, P., Barroca, L., Taylor, K., Salah, D., Sharp, H.: Agile challenges in practice: a thematic analysis, vol. 212 (2015) [115].
P44: Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations: a systematic literature review. Journal of Systems and Software **119**, 87–108 (2016) [116].
P45: Eloranta, V.P., Koskimies, K., Mikkonen, T.: Exploring ScrumBut - an empirical study of scrum anti-patterns. Information and Software Technology **74**, 194–203 (2016) [69].
P46: Uikey, N., Suman, U.: Tailoring for agile methodologies: a framework for sustaining quality and productivity. International Journal of Business Information Systems **23**(4), 432–455 (2016) [56].
P46: Tripp, J., Armstrong, D.: Agile methodologies: organizational adoption motives, tailoring, and performance. Journal of Computer Information Systems **58**, 1–10 (10 2016). [59].
P48: Kropp, M., Meier, A., Biddle, R.: Agile practices, collaboration and experience an empirical study about the effect of experience in agile software development, vol. 10027 LNCS (2016) [57].
P49: Jain, R., Suman, U.: Effectiveness of agile practices in global software development. International Journal of Grid and Distributed Computing **9**(10), 231–248 (2016) [58].
P50: Alliance, S.: Learn about scrum. https://www.scrumalliance.-org/why-scrum (2016), accessed: 2017-12-01 [8].
P51: Hoda, R., Noble, J.: Becoming agile: a grounded theory of agile transitions in practice. In: Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017. pp. 141–151 (2017) [117].
P52: Sunner, D.: Agile: adapting to need of the hour: understanding agile methodology and agile techniques. In: Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016. pp. 130–135 (2017) [52].
P53: Bello, M., Ghani, I.: A survey on success factors and obstacles for further adoption of agile in it organisations. International Journal of Advanced Media and Communication **7**(3), 167–180 (2017) [53].
P54: Lacerda, L., Furtado, F.: Factors that help in the implantation of agile methods: a systematic mapping of the liteature. In: Iberian Conference on Information Systems and Technologies, CISTI. vol. 2018-June, pp. 1–6 (2018) [60].

P55: Vallon, R., da Silva Estacio, B., Prikladnicki, R., Grechenig, T.: Systematic literature review on agile practices in global software development. Information and Software Technology **96**, 161–180 (2018) [61].

REFERENCES

[1] VersionOne. (2019). *The 13th Annual State of Agile Report 2019*. Accessed: Jan. 15, 2020. [Online]. Available: http://stateofagile.versionone.com

[2] E. Papatheocharous and A. S. Andreou, "Empirical evidence and state of practice of software agile teams," *J. Softw., Evol. Process*, vol. 26, no. 9, pp. 855–866, Sep. 2014.

[3] O. Ozcan-Top and O. Demirörs, "A reference model for software agility assessment: AgilityMod," in *Software Process Improvement and Capability Determination*, T. Rout, R. V. O'Connor, and A. Dorling, Eds. Cham, Switzerland: Springer, 2015, pp. 145–158.

[4] S. W. Ambler and M. Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*. Indianapolis, IN, USA: IBM Press, 2012.

[5] S. Ambler. (2018). *IT Project Success Rates Survey Results*. Accessed: Oct. 1, 2019. [Online]. Available: http://www.ambysoft.com/surveys/success2018.html

[6] O. E. Adalı, Ö. Özcan-Top, and O. Demirörs, "Evaluation of agility assessment tools: A multiple case study," in *Software Process Improvement and Capability Determination*, P. M. Clarke, R. V. O'Connor, T. Rout, and A. Dorling, Eds. Cham, Switzerland: Springer, 2016, pp. 135–149.

[7] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. (2001). *Manifesto for agile software development*. [Online]. Available: http://www.agilemanifesto.org/

[8] S. Alliance. (2016). *Learn About Scrum*. Accessed: Dec. 1, 2017. [Online]. Available: https://www.scrumalliance.-org/why-scrum

[9] W. Cunningham. (1999). *Extreme Programming*. Accessed: Dec. 1, 2017. [Online]. Available: http://www.-extremeprogramming.org/

[10] A. Cockburn. *Agile Software Development*. Reading, MA, USA: Addison-Wesley, 2002.

[11] E. Foundation. (2012). *OpenUp Methodology*. Accessed: Dec. 1, 2017. [Online]. Available: http://epf.eclipse.org/wikis/openup/

[12] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, *Refactoring: Improving the Design of Existing Code*. Reading, MA, USA: Addison-Wesley, 1999.

[13] U. Telemaco, T. Oliveira, P. Alencar, and D. Cowan, "A catalog of bad agile smells for agility assessment," in *Proc. Ibero-Amer. Conf. Softw. Eng. (CIbSE)*, Havana, Cuba, 2019, pp. 30–43.

[14] K. Beck, "Embracing change with extreme programming," *Computer*, vol. 32, no. 10, pp. 70–77, 1999.

[15] K. Beck, *Extreme Programming Explained: Embrace Change*. Boston, MA, USA: Addison-Wesley, 2000.

[16] K. Schwaber, "SCRUM development process," in *Business Object Design and Implementation*. London, U.K.: Springer, 1997, pp. 117–134.

[17] K. Schwaber and M. Beedle, *Agile Software Development With Scrum*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[18] M. Berteig. (2015). *Rules of Scrum*. Accessed: Dec. 1, 2017. [Online]. Available: http://www.agileadvice.com/rules-of-scrum/

[19] S. R. Palmer and M. Felsing, *A Practical Guide to Feature-Driven Development*, 1st ed. London, U.K.: Pearson, 2001.

[20] J. D. Luca. (1999). *Feature Driven Development FDD*. Accessed: Dec. 1, 2017. [Online]. Available: http://www.-featuredrivendevelopment.com/

[21] J. Stapleton, *Dynamic Systems Development Method: The Method in Practice*. Boston, MA, USA: Addison-Wesley, 1997.

[22] J. A. Highsmith, III, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. New York, NY, USA: Dorset House, 2000.

[23] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods—Review and analysis," VTT Publications, Espoo, Finland, Tech. Rep. 478, 2002.

[24] S. Fraser, B. Boehm, J. Järkvik, E. Lundh, and K. Vilkki, "How do Agile/XP development methods affect companies?" in *Extreme Programming and Agile Processes in Software Engineering*, P. Abrahamsson, M. Marchesi, and G. Succi, Eds. Berlin, Germany: Springer, 2006, pp. 225–228.

[25] T. J. Gandomani, H. Zulzalil, A. A. A. Ghani, A. B. M. Sultan, and M. Z. Nafchi, "Obstacles in moving to agile software development methods at a glance," *J. Comput. Sci.*, vol. 9, no. 5, p. 620 2013.

[26] T. J. Gandomani, H. Zulzalil, A. Ghani, A. Azim, and A. B. Sultan, "Important considerations for agile software development methods governance," *J. Theor. Appl. Inf. Technol.*, vol. 55, no. 3, pp. 345–351, 2013.

[27] S. Soundararajan and J. D. Arthur, "A structured framework for assessing the' goodness' of agile methods," in *Proc. 18th IEEE Int. Conf. Workshops Eng. Comput.-Based Syst.*, Apr. 2011, pp. 14–23.

[28] W. H. Brown, R. C. Malveau, H. W. McCormick, and T. J. Mowbray, *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. Hoboken, NJ, USA: Wiley, 1998.

[29] E. van Emden and L. Moonen, "Java quality assurance by detecting code smells," in *Proc. 9th Work. Conf. Reverse Eng.*, 2002, pp. 97–106.

[30] M. Lanza and R. Marinescu, *Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems*. Berlin, Germany: Springer-Verlag, 2007.

[31] G. Travassos, F. Shull, M. Fredericks, and V. R. Basili, "Detecting defects in object-oriented designs: Using reading techniques to increase software quality," *ACM SIGPLAN Notices*, vol. 34, no. 10, pp. 47–56, Oct. 1999.

[32] N. Moha, Y.-G. Gueheneuc, L. Duchien, and A.-F. Le Meur, "DECOR: A method for the specification and detection of code and design smells," *IEEE Trans. Softw. Eng.*, vol. 36, no. 1, pp. 20–36, Jan. 2010.

[33] J. Schumacher, N. Zazworka, F. Shull, C. Seaman, and M. Shaw, "Building empirical support for automated code smell detection," in *Proc. ACM-IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Sep. 2010, pp. 1–10.

[34] F. Arcelli Fontana, P. Braione, and M. Zanoni, "Automatic detection of bad smells in code: An experimental assessment.," *J. Object Technol.*, vol. 11, no. 2, p. 5:1, 2012.

[35] R. O. Spinola, A. C. Dias-Neto, and G. H. Travassos, "Abordagem para desenvolver tecnologia de software com apoio de estudos secundários e primários," in *Proc. Exp. Softw. Eng. Latin Amer. Workshop (ESELAW)*, 2008, pp. 1–25.

[36] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele Univ., Univ. Durham, Durham, U.K., Tech. Rep. EBSE 2007-001, Jul. 2007.

[37] M. Pai, M. McCulloch, J. Gorman, N. Pai, W. Enanoria, G. Kennedy, P. Tharyan, and J. Colford, "Systematic reviews and meta-analyses: An illustrated, step-by-step guide," *Med. J. India*, vol. 17, no. 2, pp. 86–95, 2004.

[38] G. G. Miller, "The characteristics of agile software processes," in *Proc. 39th Int. Conf. Exhib. Technol. Object-Oriented Lang. Syst. (TOOLS)*. Washington, DC, USA: IEEE Computer Society, 2001.

[39] M. Lindvall, V. R. Basili, B. W. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. A. Williams, and M. V. Zelkowitz, "Empirical findings in agile methods," in *Proc. 2nd XP Universe 1st Agile Universe Conf. Extreme Program. Agile Methods*. London, U.K.: Springer-Verlag, 2002.

[40] J. F. Abrantes and G. H. Travassos, "Common agile practices in software processes," in *Proc. Int. Symp. Empirical Softw. Eng. Meas.*, Sep. 2011, pp. 355–358.

[41] N. C. Tsourveloudis and K. P. Valavanis, "On the measurement of enterprise agility," *J. Intell. Robotic Syst.*, vol. 33, no. 3, 329–342, 2002.

[42] A. Gill and B. Henderson-Sellers, "Measuring agility and adaptibility of agile methods: A 4 dimensional analytical tool," in *Proc. Int. Conf. Appl. Comput. (IADIS)*. Lisbon, Portugal: IADIS Press, 2006.

[43] J. Li, J. F. Burnham, T. Lemley, and R. M. Britton, "Citation analysis: Comparison of Web of science, scopus, SciFinder, and Google scholar," *J. Electron. Resour. Med. libraries*, vol. 7, no. 3, pp. 196–217, 2010.

[44] F. Maurer and S. Martel, "Extreme programming. Rapid development for Web-based applications," *IEEE Internet Comput.*, vol. 6, no. 1, pp. 86–90, 2002.

[45] J. Newkirk, "Introduction to agile processes and extreme programming," in *Proc. 24th Int. Conf. Softw. Eng. (ICSE)*, May 2002, pp. 695–696.

[46] R. C. Martin, *Agile Software Development: Principles, Patterns, and Practices*. Upper Saddle River, NJ, USA: Prentice-Hall, 2003.

[47] L. Williams, "Agile software development methodologies and practices," *Adv. Comput.*, vol. 80, pp. 1–44, Jan. 2010.

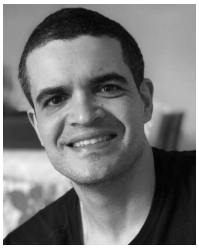
[48] Z. Alzoabi, "Agile software: Body of knowledge," in *Business Intelligence and Agile Methodologies for Knowledge-Based Organizations: Cross-Disciplinary Applications*. Hershey, PA, USA: IGI Global, 2012, pp. 14–34.

[49] A. Moran, "Agile software development," in *Agile Risk Management*. Cham, Switzerland: Springer, 2014, pp. 1–16.

[50] P. Diebold and M. Dahlem, "Agile practices in practice: A mapping study," in *Proc. 18th Int. Conf. Eval. Assessment Softw. Eng.*, 2014, pp. 1–10.

[51] I. Ghani and M. Bello, "Agile adoption in IT organizations," *KSII Trans. Internet Inf. Syst.*, vol. 9, no. 8, pp. 3231–3248, 2015.
[52] D. Sunner, "Agile: Adapting to need of the hour: Understanding agile methodology and agile techniques," in *Proc. 2nd Int. Conf. Appl. Theor. Comput. Commun. Technol. (iCATccT)*, 2016, pp. 130–135.
[53] M. Bello and I. Ghani, "A survey on success factors and obstacles for further adoption of agile in IT organisations," *Int. J. Adv. Media Commun.*, vol. 7, no. 3, pp. 167–180, 2017.
[54] M. T. Sletholt, J. E. Hannay, D. Pfahl, and H. P. Langtangen, "What do we know about scientific software Development's agile practices?" *Comput. Sci. Eng.*, vol. 14, no. 2, pp. 24–37, Mar. 2012.
[55] B. Meyer, *Agile!: The Good, the Hype and the Ugly*. Cham, Switzerland: Springer, 2014.
[56] U. Suman and N. Uikey, "Tailoring for agile methodologies: A framework for sustaining quality and productivity," *Int. J. Bus. Inf. Syst.*, vol. 23, no. 4, pp. 432–455, 2016.
[57] M. Kropp, A. Meier, and R. Biddle, "Agile practices, collaboration and experience," in *Product-Focused Software Process Improvement*, P. Abrahamsson, A. Jedlitschka, A. N. Duc, M. Felderer, S. Amasaki, and T. Mikkonen, Eds. Cham, Switzerland: Springer, 2016, pp. 416–431.
[58] R. Jain and U. Suman, "Effectiveness of agile practices in global software development," *Int. J. Grid Distrib. Comput.*, vol. 9, no. 10, pp. 231–248, Oct. 2016.
[59] J. F. Tripp and D. J. Armstrong, "Agile methodologies: Organizational adoption motives, tailoring, and performance," *J. Comput. Inf. Syst.*, vol. 58, no. 2, pp. 170–179, Apr. 2018, doi: 10.1080/08874417.2016.1220240.
[60] L. L. Lacerda and F. Furtado, "Factors that help in the implantation of agile methods: A systematic mapping of the liteature," in *Proc. 13th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2018, pp. 1–6.
[61] R. Vallon, B. J. da Silva Estácio, R. Prikladnicki, and T. Grechenig, "Systematic literature review on agile practices in global software development," *Inf. Softw. Technol.*, vol. 96, pp. 161–180, Apr. 2018.
[62] T. H. Muhammad Faisal Nisar, "Agile methods handling offshore software development issues," in *Proc. 8th Int. Multitopic Conf. (INMIC)*, 2004, pp. 417–422, doi: 10.1109/INMIC.2004.1492915.
[63] E. Yatzeck. (Dec. 2012). *A Corporate Agile 10-Point Checklist*. Accessed: Jun. 30, 2019. [Online]. Available: http://pagilista.blogspot.com/2012/12/a-corporate-agile-10-point-checklist.html
[64] L. F. Chagas, D. D. de Carvalho, A. M. Lima, and C. A. L. Reis, "Systematic literature review on the characteristics of agile project management in the context of maturity models," in *Software Process Improvement and Capability Determination*. Cham, Switzerland: Springer, 2014, pp. 177–189.
[65] L. Cao and B. Ramesh, "Agile software development: Ad hoc practices or sound principles?" *IT Prof.*, vol. 9, no. 2, pp. 41–47, Mar. 2007.
[66] J. Thomas, "Introducing agile development practices from the middle," in *Proc. 15th Annu. IEEE Int. Conf. Workshop Eng. Comput. Based Syst. (ECBS)*, Mar. 2008, pp. 401–407.
[67] S. C. Misra, V. Kumar, and U. Kumar, "Identifying some important success factors in adopting agile software development practices," *J. Syst. Softw.*, vol. 82, no. 11, pp. 1869–1890, Nov. 2009.
[68] J. Li and X. Wang, "Research and practice of agile unified process," in *Proc. 2nd Int. Conf. Softw. Technol. Eng.*, vol. 2, Oct. 2010, pp. V2340–V2343.
[69] V.-P. Eloranta, K. Koskimies, and T. Mikkonen, "Exploring ScrumBut—An empirical study of scrum anti-patterns," *Inf. Softw. Technol.*, vol. 74, pp. 194–203, Jun. 2016.
[70] P. McMahon, "Extending agile methods: A distributed project and organizational improvement perspective," *CrossTalk*, vol. 5, pp. 16–19, Apr. 2005.
[71] S. Zhong, C. Liping, and C. Tian-En, "Agile planning and development methods," in *Proc. 3rd Int. Conf. Comput. Res. Develop.*, vol. 1, Mar. 2011, pp. 488–491.
[72] A. B. Kajornboon, "Using interviews as research instruments," *E-J. Res. Teachers*, vol. 2, no. 1, pp. 1–9, 2005.
[73] S. Oishi, *How to Conduct in-Person Interviews for Surveys*. Newcastle upon Tyne, U.K.: SAGE, 2003.
[74] A. N. Ghazi, K. Petersen, S. S. V. R. Reddy, and H. Nekkanti, "Survey research in software engineering: Problems and strategies," 2017, *arXiv:1704.01090*. [Online]. Available: http://arxiv.org/abs/1704.01090
[75] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. London, U.K.: Pearson, 1995.
[76] P. Cohen, J. Cohen, L. S. Aiken, and S. G. West, "The problem of units and the circumstance for POMP," *Multivariate Behav. Res.*, vol. 34, no. 3, pp. 315–346, Jul. 1999.
[77] A. S. Sidky, "A structured approach to adopting agile practices: The agile adoption framework," Ph.D. dissertation, Virginia Polytech. Inst., State Univ., Blacksburg, VI, USA, 2007.
[78] J. Shore and S. Warden, *The Art of Agile Development: Pragmatic Guide to Agile Software Development*. Newton, MA, USA: O'Reilly Media, 2007.
[79] V. Subramaniam and A. Hunt, *Practices of an Agile Developer: Working in the Real World*. Raleigh, NC, USA: Pragmatic Bookshelf, 2006.
[80] O. Ktata and G. Lévesque, "Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects," in *Proc. 2nd Canadian Conf. Comput. Sci. Softw. Eng.* New York, NY, USA: ACM, 2009, pp. 59–66.
[81] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies," *J. Syst. Softw.*, vol. 85, no. 6, 1213–1221, Jun. 2012.
[82] X. Yu and S. Petter, "Understanding agile software development practices using shared mental models theory," *Inf. Softw. Technol.*, vol. 56, no. 8, pp. 911–921, Aug. 2014.
[83] S. Hermida. (2009). *A Better Team*. Accessed: Jun. 30, 2019. [Online]. Available: http://www.abetterteam.org/
[84] W. Krebs, P. Morgan, and R. Ashton. (2011). *Agile Journey Index*. AgileBill Krebs. Accessed: Jun. 30, 2019. [Online]. Available: http://www.agiledimensions.com/blog/wp-content/uploads/2011/10/KrebsAgileJourneyIndex.pdf
[85] L. Williams, K. Rubin, and M. Cohn, "Driving process improvement via comparative agility assessment," in *Proc. Agile Conf.*, Aug. 2010, pp. 3–10.
[86] D. MacKeen. (2013). *Enterprise and Team Level Agility Maturity Matrix*. Eliassen Group. Accessed: Jun. 30, 2019. [Online]. Available: http://blog.eliassen.com/introducing-the-enterprise-agility-maturity-matrix
[87] *IBM: IBM Devops Self-Assessment*. Accessed: Jun. 30, 2019. [Online]. Available: https://devopsassessment.mybluemix.net/#/
[88] H. Kniberg. (2012). *Scrum Checklist*. Accessed: Jun. 30, 2019. [Online]. Available: http://www.crisp.se/scrum/checklist
[89] B. Campbell and R. MacIver. (2010). *Agility Maturity Self Assessment*. Accessed: Jun. 30, 2019. [Online]. Available: http://www.robbiemaciver.com/documents/presentations/A2010-Agile%20Maturity%20Self-Assessment.pdf
[90] E. Ribeiro. (2015). *Agility Maturity Self Assessment Survey*. Accessed: Jun. 30, 2019. [Online]. Available: https://beyondleanagile.com/2015/12/08/agile-maturity-self-assessment-survey-published-at-scrumalliance/
[91] D. Tousignant. (2013). *How Agile Are You? Free Agile Maturity Assessment*. Accessed: Jun. 30, 2019. [Online]. Available: https://capeprojectmanagement.com/agile-self-assessment/
[92] K. Waters. (2008). *How Agile Are You? Free Agile Maturity Assessment*. Accessed: Jun. 30, 2019. [Online]. Available: https://www.101ways.com/2008/01/21/how-agile-are-you-take-this-42-point-test/
[93] J. Rothman. (2013). *Self Assessment Tool for Transitioning to Agile*. Accessed: Jun. 30, 2019. [Online]. Available: https://www.jrothman.com/mpd/agile/2013/04/self-assessment-tool-for-transitioning-to-agile/
[94] O. R. Yürüm, O. Demirörs, and F. Rabhi, "A comprehensive evaluation of agile maturity self-assessment surveys," in *Software Process Improvement and Capability Determination*, I. Stamelos, R. V. O'Connor, T. Rout, and A. Dorling, Eds. Cham, Switzerland: Springer, 2018, pp. 300–315.
[95] Borland. (2009). *Borland Agile Assessment*. Accessed: Jun. 30, 2019. [Online]. Available: https://borland.typepad.com/agile-transformation/2009/03/borland-agile-assessment-2009.html
[96] I. T. R. Group. (2013). *Agile Process Assessment Tool*. Accessed: Jun. 30, 2019. [Online]. Available: https://www.infotech.com/research/ss/it-deploy-changes-more-rapidly-by-going-agile/it-agile-process-assessment-tool
[97] E. Gunnerson. (2015). *Agile Team Evaluation*. Accessed: Jun. 30, 2019. [Online]. Available: https://blogs.msdn.microsoft.com/ericgu/2015/10/05/agile-team-evaluation/
[98] (2015). *VersionOne: Agile Assessment: Test Your Team's Agility*. [Online]. Available: https://resources.collab.net/agile-101/agile-assessment, accessed: 2019-06-30
[99] M. Britsch. (2017). *Agility Questionnaire*. Accessed: Jun. 30, 2019. [Online]. Available: https://thedigitalbusinessanalyst.co.uk/agility-questionnaire-130b03133b98
[100] M. Sahota. (2010). *An Agile Adoption and Transformation Survival Guide*. [Online]. Available: http://www.barryovereem.com/wp-content/uploads/M.-Sahota-Checklist-for-Change-Agents.pdf
[101] K. Chronis "Measuring agility—A validity study on tools measuring the agility level of software development teams," M.S. thesis, Univ. Gotherburg, Goteborg, Sweden, 2015.

[102] D. Leffingwell, *Scaling Software Agility: Best Practices for Large Enterprises*. London, U.K.: Pearson, 2007.
[103] C. So and W. Scholl, "Perceptive agile measurement: New instruments for quantitative studies in the pursuit of the social-psychological effect of agile practices," in *Agile Processes in Software Engineering and Extreme Programming*, P. Abrahamsson, M. Marchesi, and F. Maurer, Eds. Berlin, Germany: Springer, 2009, pp. 83–93.
[104] S. Soundararajan, "Assessing agile methods: Investigating adequacy, capability, and effectiveness an objectives, principles, strategies approach," Ph.D. dissertation, Virginia Polytech. Inst. State Univ., Blacksburg, VI, USA, 2013.
[105] K. Nebe and S. Baloni, "Agile human-centred design: A conformance checklist," in *Human Interface and the Management of Information: Information, Design and Interaction*, S. Yamamoto, Ed. Cham, Switzerland: Springer, 2016, pp. 442–453.
[106] A. Qumer and B. Henderson-Sellers, "A framework to support the evaluation, adoption and improvement of agile methods in practice," *J. Syst. Softw.*, vol. 81, no. 11, pp. 1899–1919, Nov. 2008.
[107] R. L. Glass and T. DeMarco, *Software Creativity 2.0*. Atlanta, GA, USA: Developer Dot Star Books, 2006.
[108] Miller, G.:, "Agile software development for the entire project," *CrossTalk*, vol. 18, no. 12, pp. 9–12, 2005.
[109] M. Pikkarainen, O. Salo, and J. Still, "Deploying agile practices in organizations: A case study," in *Software Process Improvement*, I. Richardson, P. Abrahamsson, and R. Messnarz, Eds. Berlin, Germany: Springer, 2005, pp. 16–27.
[110] S. Ambler, "Survey says: Agile works in practice," *Dobb's J.*, vol. 31, no. 9, pp.62–64, 2006.
[111] K. Kautz, C. Pedersen, and O. Monrad, "Cultures of agility—Agile software development in practice," in *Proc. 20th Australas. Conf. Inf. Syst. (ACIS)*, 2009, pp. 174–184.
[112] D. Batra, "Modified agile practices for outsourced software projects," *Commun. ACM*, vol. 52, no. 9, pp. 143–148 and 10, Sep. 2009.
[113] M. Poppendieck and M. A. Cusumano, "Lean software development: A tutorial," *IEEE Softw.*, vol. 29, no. 5, pp. 26–32, Sep. 2012.
[114] S. Dyck and T. A. Majchrzak, "Identifying common characteristics in fundamental, integrated, and agile software development methodologies," in *Proc. 45th Hawaii Int. Conf. Syst. Sci.*, Jan. 2012, pp. 5299–5308.
[115] P. Gregory, L. Barroca, K. Taylor, D. Salah, and H. Sharp, "Agile challenges in practice: A thematic analysis," in *Proc. Int. Conf. Agile Softw. Develop.*, vol. 212, 2015, pp. 64–80.
[116] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review," *J. Syst. Softw.*, vol. 119, pp. 87–108, Sep. 2016.
[117] R. Hoda and J. Noble, "Becoming agile: A grounded theory of agile transitions in practice," in *Proc. IEEE/ACM 39th Int. Conf. Softw. Eng. (ICSE)*, May 2017, pp. 141–151.



ULISSES TELEMACO received the bachelor's degree in computer science from the Federal University of Rio Grande do Norte, in 2002 and the M.Sc. degree in mechatronic engineering from the Federal University of Bahia in 2009. He is currently pursuing the Ph.D. degree in software engineering with the Federal University of Rio de Janeiro (UFRJ). He is also a Visiting Researcher with the University of Waterloo, Canada. He is also a Chief Technology Officer (CTO) at OWSE Web Software Engineering, Inc., an Experienced Developer with more than 20 years working as a Software Architect. As a Technology Evangelist, he has actively participated in the promotion of Java User Groups across Brazil and around the world.

TOACY OLIVEIRA received the degree in electrical engineering, M.Sc., and Ph.D. degrees from the Pontifical Catholic University of Rio de Janeiro, Brazil, in 1992, 1997, and 2001, respectively. He spent three years at the University of Waterloo as a Postdoctoral Fellow. He is currently an Assistant Professor with the Federal University of Rio de Janeiro with the Systems and Computing Engineering Program, COPPE/UFRJ, Brazil. He is also an Adjunct Professor with the David R. Cheriton School of Computer Science, University of Waterloo, Canada. He has published more than 70 refereed publications, and has been a member of program committees of numerous conferences and workshops. Besides being a leading investigator in Brazilian research projects supported by CAPES and CNPq, he has also exercised entrepreneurship by founding several companies in Brazil. His current research interests focus on harnessing knowledge intensive processes such those realized as modern software systems. His research attempts to envision new techniques to capture, assess, and implement such type of processes.

PAULO ALENCAR is currently a Research Professor with the David R. Cheriton School of Computer Science, University of Waterloo, and also an Associate Director of the Computer Systems Group (CSG). He has received international research awards from organizations such as Compaq and IBM. He has published more than 200 refereed publications and has been a member of program committees of numerous highly-regarded conferences and workshops. His recent research in information technology and software engineering has focused on high-level software architectures, design, components, and their interfaces; software frameworks and application families; software processes, automated workflows and work graphs, and evolution; Web-based approaches and applications; open and big data applications; context-aware and event-based systems; software agents; machine learning; cognitive chatbots; artificial intelligence; and formal methods. He has been a Principal or a Co-Principal Investigator in projects supported by NSERC, ORF-RE, IBM, SAP, CITO, CSER, Bell, and funding agencies in Canada U.S., Brazil, Germany, and Argentina. He is a member of the Association of Computing Machinery (ACM), the Institute of Electrical and Electronic Engineers (IEEE), the Association for the Advancement of Artificial Intelligence (AAAI), the Waterloo Water Institute (part of the Global Water Futures initiative), and the Waterloo Artificial Intelligence Institute.

DON COWAN is currently a Distinguished Professor Emeritus of computer science with the University of Waterloo and also the Director of the Computer Systems Group. He has made contributions to computer science in areas such as computer science, software engineering, and complex applications. He has authored or coauthored or an editor of more than 300 refereed articles and 17 books in computer/communications, software engineering, education, environmental information systems, and mathematics. He has supervised more than 120 graduate students and postdoctoral fellows. His group has developed over 80 web-based and mobile software systems for many applications in areas such as volunteerism, environment, socioeconomic development, tourist, population health, aboriginal affairs, arts and culture, and built heritage. The University of Waterloo recognized his contributions to development of graduate students by presenting him with the Award of Excellence in Graduate Supervision. He was recognized for his research and support of the development of computer science in Brazil by being awarded the National Order of Scientific Merit (Grand Cross), the country's highest civilian scientific honor, by the President of Brazil. In 2009, he received the Waterloo Award, the City of Waterloo's highest civic honour, for his contributions to the City of Waterloo. In 2010, he was named a Distinguished Scientist by the Association for Computing Machinery and, in 2011, he received a Doctor of Science (Honoris Causa) from the University of Guelph for his contributions to Computer Science and Software Engineering.