Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

COPPE UFRJ

# AGILEQUBE: AN APPROACH FOR SPECIFICATION AND DETECTION OF AGILE SMELLS

Ulisses Telemaco

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Toacy Oliveira

Rio de Janeiro
Dezembro de 2020

# AGILEQUBE: AN APPROACH FOR SPECIFICATION AND DETECTION OF AGILE SMELLS

Ulisses Telemaco

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Toacy Oliveira

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2020

Telemaco, Ulisses

AgileQube: An approach for specification and detection of agile smells/Ulisses Telemaco. – Rio de Janeiro: UFRJ/COPPE, 2020.

XIX, 237 p. 29, 7cm.

Orientador: Toacy Oliveira

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2020.

Bibliography: p. 143 – 165.

1. Agility Assessment. 2. Agile Assessment. 3. Agile Smell. I. Oliveira, Toacy. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

*Às mulheres da minha vida:*
*Noilza, Vanessa, Laura e Beatriz.*

# Agradecimentos

Muita gente me ajudou nessa longa jornada, portanto, é um grande desafio escrever essa seção. Mas gostaria de agradecer primeiramente a Deus por Ele estar sempre ao meu lado e por ter me ajudado a tomar as decisões certas que me fizeram chegar até aqui.

Agradeço imensamente a Vanessa, minha esposa e cara-metade. Não tenho palavras para dizer o quanto sou grato por ela ter comprado essa ideia, pelos muitos sacríficios que ela teve que fazer e por ter sido a retaguarda de apoio que nos permitiu chegar até aqui. É injusto ela não ganhar oficialmente um título. Mas gostaria que ela soubesse que ela estará sempre homenageada em meu coração. Agradeço também às minhas filhas, Laura e Beatriz, porque vocês foram meu maior incentivo nessa jornada. Espero que essa experiência inspire as suas vidas e que possam conquistar tudo aquilo que desejarem.

A todos de minha família. Em especial à minha mãe, Noilza, por ter sido de fato a primeira pessoa a participar dessa jornada, por me amar e acreditar em mim (mais do eu mesmo) e por ter vibrado com o Doutorado desde o princípio. Ao meu pai, Telemaco, por ter me ensinado princípios e valores que levarei por toda a vida por ter sido um exemplo de pai e de profissional. A sua figura me inspirou e continua me inspirando muito. Às minhas irmãs, Emmanuelle e Caroline, por termos crescidos unidos e com muito amor. Esses são, sem dúvidas, ingredientes catalisadores de grandes transformações e conquistas como um doutorado. Ao meu tio Dr. Isaias Paiva, por ter sido sempre uma pessoa tão amiga e pelo profissional excepcional que ele é. O seu amor pela medicina, pela pesquisa científica, pelas pessoas ao seu redor e a sua inteligência são fontes de inspiração para mim.

A meu orientador, Prof. Toacy Oliveira, por ter acreditado em mim desde o princípio e pelas orientações durantes os últimos anos. Sem a sua colaboração, confiança, paciência e persistência, nunca teria sido possível concluir esta dissertação.

Aos Professores Paulo Alencar e Don Cowan por terem proporcionado a mim e a minha família a oportunidade de vir para o Canadá e fazer parte de uma instituição

de grande prestígio como a *University of Waterloo*. Espero que a nossa parceria seja cada vez mais sólida.

Aos meus amigos do grupo de pesquisa Prisma, Renata, Rachel, Gláucia, Edson, Fábio e Raquel. Gostaria de agradecê-los por terem me ajudado, de diversas formas, nessa dura jornada. Sem vocês, tudo teria sido ainda mais difícil.

Aos meus sócios e colegas de trabalho da OWSE que direta ou indiretamente contribuíram para a realização deste trabalho, em especial ao meu amigo Walter Magioli por ter sido um grande incetivador e entusiasta do meu doutorado.

Aos professores da linha de Engenharia de Software do PESC, em especial aos professores Guilherme Travassos, Cláudia Werner e Geraldo Xexeo pela contribuição ao meu aprendizado e formação como pesquisador. Meu muito obrigado também aos funcionários da secretaria do PESC, pela dedicação ao trabalho de suporte aos alunos do programa, especialmente ao Gutierrez da Costa, Ricardo César e Solange.

Ao *Emerging Leaders in the Americas Program* (ELAP) e ao Governo Canadense por terem viabilizado a minha vinda para o Canadá que foi fundamental para essa pesquisa.

E para finalizar, a todas as pessoas que participaram do *survey* conduzido durante esse estudo.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

AGILEQUBE: UMA ABORDAGEM PARA ESPECIFICAÇÃO DE DETECÇÃO DE AGILE SMELLS

Ulisses Telemaco

Dezembro/2020

Orientador: Toacy Oliveira

Programa: Engenharia de Sistemas e Computação

**Contexto**: Nos últimos anos, o Desenvolvimento Ágil (AD) tem sido um dos tópicos mais importantes para a comunidade de Engenharia de Software e tema de diversos estudos acadêmicos e iniciativas da indústria. Mas “Ser Ágil” se tornou mais do que uma tendência entre a comunidade de desenvolvimento de software e, atualmente, é um movimento estratégico para manter as empresas competitivas. No entanto, apesar de todo o esforço que as empresas investem para entender como adotar práticas ágeis de forma eficaz, o número de empresas que de fato dominam o desenvolvimento ágil é baixo. Nesse contexto, a Avaliação da Agilidade (AA), ou seja, a investigação de como uma empresa adota práticas ágeis, é uma importante ferramenta para auxiliar organizações na adoção do desenvolvimento ágil.

**Problemas e Objetivos**: Nesta pesquisa, pretendemos contribuir para a área de AA da seguinte forma: primeiro, investigamos as abordagens de AA existentes na indústria e na academia e identificamos os seguintes problemas: 1. *Critérios de avaliação não explícitos*; 2. *Falta de um mecanismo para representação dos critérios de avaliação*; 3. *Falta de um suporte para inclusão de novos critérios de avaliação*; 4. *Coleta e entrada de dados predominantemente manual*; 5. *Falta de feedback em tempo real*; and 6. *Escalabilidade limitada*. Para resolver esses problemas, estendemos o termo *code smell* para o contexto de avaliação de agilidade, introduzimos a metáfora *agile smell* para denotar uma situação que em pode prejudicar a adoção de uma prática ágil e propusemos uma abordagem de avaliação de agilidade baseada na detecção automática (ou semi-automática) de *agile smells* em projetos ágeis.

**Metodologia**: Esta pesquisa foi organizada em 4 fases conforme a metodologia proposta por PEFFERS *et al.* 2007 [1]. Na fase 1 (*Identificação dos problemas e definição dos objetos da pesquisa*), identificamos os problemas acima mencionados e definimos os objetivos da pesquisa. Na fase 2 (*Projeto e desenvolvimento da solução*), realizamos uma revisão da literatura para identificar um conjunto de *agile smells* e um *survey* com profissionais da indústria para entender a relevância dos *agile smells* identificados. Outra revisão da literatura foi conduzida para investigar como os metamodelos para representação de processos de software existentes podem ser usados para representar um projeto ágil. Na fase 3 (*Demonstração e Avaliação*), conduzimos 2 estudos de caso para validar a abordagem proposta. Na fase 4 (*Comunicação*), publicamos 3 estudos para comunicar alguns resultados preliminares desta pesquisa.

**Resultados**: Esta pesquisa produziu as seguintes contribuições: *(a) Catalogue of Agile Smells*, um catálogo com 20 *agile smells* que serve como base para a abordagem proposta.; *(b) Agile Project Metamodel*, um metamodelo que contém os elementos necessários para representar um projeto ágil; *(c) Agile Smell Schema*, um *schema* usado para especificar os *agile smells*; e *(d) AgileQube App*, uma infraestrutura de suporte computacional formada por 4 elementos (*Specification Module*, *ETL Module*, *Detection Engine* e *Validation Module*) que suportam a especificação e detecção de *agile smells* em projetos ágeis.

**Conclusão**: As contribuições dessa pesquisa mitigaram os problemas identificados nessa pesquisa e os resultados observados nos estudos de caso confirmam que a abordagem proposta foi capaz de detectar *agile smells* de forma automática nos projetos ágeis avaliados.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

AGILEQUBE: AN APPROACH FOR SPECIFICATION AND DETECTION OF AGILE SMELLS

Ulisses Telemaco

December/2020

Advisor: Toacy Oliveira

Department: Systems Engineering and Computer Science

**Background**: Over the last years, Agile Development (AD) has been one of the most important topics for the Software Engineering community and the subject of several academic studies and industry initiatives. "Being Agile" became more than a trend among the geek community and, nowadays, it is a strategic move that the software industry has embraced and that is critical to keeping companies in the game in such a competitive industry. However, despite the effort spent to understand how companies can adopt agile practices more effectively, the number of companies that master agile development is low. In this context, agility assessment, i.e., an investigation on how a company is adopting agile practices, is one of the tools to aid the agile development adoption.

**Problems & Goals**: In this research, we aimed to contribute to the area of agility assessment in the following way: first, we investigated existing agility assessment approaches and identified the following problems: 1. *Unclear assessment criteria selection*; 2. *Unclear assessment criteria representation*; 3. *Lack of support for adding new assessment criterion*; 4. *Manual data collection and input*; 5. *Lack of real-time assessment feedback*; and 6. *Limited Scalability*. To address these problems, we extended the *code smell* term to the context of agility assessment, introduced the *agile smell* metaphor to denote a situation that may harm the adoption of an agile practice and proposed an agility assessment approach that automatically (or semi-automatically) detects agile smells in agile projects.

**Method**: This research was organized in 4 stages based on the methodology proposed by PEFFERS *et al.* 2007 [1]. In Stage 1 (*Identify the problem and define the objectives*), we identified the above mentioned problems and defined the research goals. In Stage 2 (*Design and development of the solution*), we conducted a literature review to identify a set of agile smells and a survey with practitioners to reveal the relevance of the identified agile smells. Another literature review was conducted to investigate how existing software process metamodels could be used to represent the data from an agile project. In Stage 3 (*Demonstration and Evaluation*), we conducted 2 case studies to validate the proposed approach. In Stage 4 (*Communication*), we published 3 studies to communicate some preliminary results of this research.

**Results**: This research produced the following contributions: *(a)* the *Catalogue of Agile Smells*, a catalogue that acts as the baseline for the proposed approach and has 20 agile smells; *(b)* the *Agile Project Metamodel*, a metamodel that contains the elements necessary to represent an agile project; *(c)* the *Agile Smell Schema*, a schema that enables the systematic specification of the agile smells; and *(d)* the *AgileQube App*, a computational supporting infrastructure composed of 4 elements (*Specification Module*, *ETL Module*, *Detection Engine*, and *Validation Module*) that together support the specification and detection of agile smells in agile projects.

**Conclusion**: The resulting contributions addressed the problems identified in the existing agility assessment approaches and the reports generated in the case studies confirmed that the proposed approach, along with the other contributions, was able to automatically detect agile smells in the assessed agile projects.

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

**AA** *AgileQube Approach*. xi, xv, 11–13, 16, 29–33, 35, 36, 55–58, 71, 72, 90, 91, 95, 98, 103–106, 109, 110, 121, 133, 135–138, 141

**APD** *Agile Project Data*. 33

**APMM** *Agile Project Metamodel*. viii, x, xii, xv, xvii, 11, 12, 15, 16, 32–35, 57–59, 68, 70–72, 76, 78, 90, 94, 97, 98, 105, 106, 108, 109, 136–141

**AQApp** *AgileQube App*. viii, x, xiii, xv, xvi, 12, 15, 16, 34, 35, 91–94, 98, 103, 104, 106, 111, 115, 116, 119, 121, 125, 131, 134, 136–140, 142

**AQL** *Agile Smell Schema*. viii, x, xii, xv, 11, 15, 16, 32–35, 72–76, 90, 93, 98, 104–106, 136–139

**AS 01** *Lower Priority Tasks Executed First*. xii, xvii, 47, 52, 60, 76, 80, 100, 104, 113, 114, 116, 119, 123, 124, 127, 128, 132, 190, 191, 199

**AS 02** *Absence of Frequent Deliveries*. xii, xvii, 43, 52, 81, 100, 114, 116–118, 124, 127, 132, 190, 191, 200

**AS 03** *Iteration Without a Deliverable*. xii, xvii, 45, 52, 82, 113, 114, 116, 124, 127, 128, 132, 190, 191, 201

**AS 04** *Goals Not Defined or Poorly Defined*. xii, xvii, 44, 52, 59, 83, 113, 114, 116, 119, 124, 127, 128, 130, 132, 190, 191, 202

**AS 05** *Iteration Without an Iteration Planning*. xii, xvii, 45, 52, 84, 114, 116, 119, 124, 127, 132, 190, 191, 203

**AS 06** *Complex Tasks*. xii, xvii, 44, 48, 52, 60, 85, 100, 104, 114, 116, 119, 124, 127, 130, 132, 190, 191, 204

**AS 07** *Iteration Without an Iteration Retrospective*. xii, xvii, 45, 52, 86, 114, 116, 119, 124, 127, 130, 132, 190, 191, 205

**AS 08** *Absence of Timeboxed Iteration*. xii, xviii, 43, 52, 60, 87, 104, 114, 116, 119, 124, 127, 130, 132, 190, 191, 206

**AS 09** *Iteration Started without an Estimated Effort*. xiii, xviii, 45, 52, 88, 114, 116, 119, 124, 127, 128, 130, 132, 190, 191, 207

**AS 10** *Iteration Without an Iteration Review*. xiii, xviii, 46, 52, 89, 100, 114, 116, 119, 123, 124, 127, 130, 132, 190, 191, 208

**AS 11** *Shared Developers*. 47, 52, 53, 190, 191

**AS 12** *Unplanned Work*. 47, 52, 53, 190, 191

**AS 13** *Dependence on Internal Specialists*. 44, 52, 190, 191

**AS 14** *Unfinished Work in a Closed Iteration*. 47, 52, 190, 191

**AS 15** *Absence of Timeboxed Meeting*. 43, 44, 52, 190, 191

**AS 15** *Absence of Test-driven Development*. 43, 52, 190, 191

**AS 17** *Large Development Team*. 46, 52, 53, 190, 191

**AS 19** *Iterations with Different Duration*. 46, 52, 190, 191

**AS 19** *Long Break Between Iterations*. 46, 52, 53, 190, 191

**AS 20** *Concurrent Iterations*. 44, 52, 190, 191

**ASRC** *Agile Smells Report*. xv, xvi, 33, 100–102, 110, 111, 116–121, 125, 126, 128–131, 134, 138

**ASRP** *Agile Smells Report (preliminary)*. 33, 98, 103, 105, 108, 121, 131, 137, 138

**ASS** *Agile Smell Specifications*. 33

**ASSC** *Agile Smell Specifications (configured)*. 33

**CAS** *Catalogue of Agile Smells*. viii, x, xi, 11, 12, 15, 16, 32, 33, 35, 36, 56, 59, 71, 77, 79, 90, 106, 123, 133, 136–138, 141, 142, 187

**DE** *Detection Engine*. viii, x, xiii, xv, 34, 37, 57, 72, 76, 79, 90, 94, 98–100, 104, 105, 108, 137, 140

**ETLM** *ETL Module*. viii, x, xiii, xv, 33, 34, 57, 94–96, 98, 104, 105, 109, 114, 115, 123, 125, 137, 140, 142

**G1** *Define a catalogue of agile smells*. xvi, 10, 11, 36, 56, 106, 135, 138, 139

**G2** *Define an agile smell representation approach*. xvi, 11, 57, 71, 72, 90, 106, 135, 139

**G3** *Define a data extraction approach*. xvi, 11, 57, 71, 135, 140

**G4** *Define and implement a computational system*. xvi, 11, 91, 105, 135, 140, 141

**GitHub** *GitHub*. xiii, xvii, 15, 96, 97, 106–109, 113, 123, 133, 134, 138, 142

**JSS** *Journal Submission System*. xiii, xiv, xvi, xvii, 15, 16, 106, 110–112, 114–121, 133, 134, 137, 211

**MSMS** Manuscrit Submission Management System. 111, 137

**P1** *Unclear assessment criteria selection*. ix, 8, 10, 29, 135, 139

**P1** *Critérios de avaliação não explícitos*. vii

**P2** *Unclear assessment criteria representation*. ix, 8, 11, 29, 135, 139

**P2** *Falta de um mecanismo para representação dos critérios de avaliação*. vii

**P3** *Lack of support for adding new assessment criterion*. ix, 9, 11, 29, 135, 139

**P3** *Falta de um suporte para inclusão de novos critérios de avaliação*. vii

**P4** *Manual data collection and input*. ix, 9, 11, 29, 135, 140

**P4** *Coleta e entrada de dados predominantemente manual*. vii

**P5** *Lack of real-time assessment feedback*. ix, 9, 11, 29, 135, 140

**P5** *Falta de feedback em tempo real*. vii

**P6** *Limited Scalability*. ix, 9, 11, 29, 135, 140

**P6** *Escalabilidade limitada*. vii

**PKD** *Prisma KIP Domain*. 12, 15

**PNM** *preconditions not met*. 108

**PZETL** *Prisma ZenHub ETL*. 12, 15, 96

**SM** *Specification Module*. viii, x, xiii, xv, 34, 92, 93, 104, 137, 139, 140

**TOS** *Terminal Operational System*. xiii, xiv, xvi–xviii, 15, 16, 106, 110, 121, 122, 124–126, 128–134, 138, 218, 219

**VM** *Validation Module*. viii, x, xiii, xvi, 34, 35, 103–105, 137, 140

**ZenHub** *ZenHub*. xiii, xv, xvii, 12, 15, 34, 94–98, 106–109, 113–116, 122, 123, 125, 133, 134, 137, 138, 142

# Chapter 1

# Introduction

*This chapter introduces the study and is organized in the following sections: Research Context, Problem Statement, Research Goals, Research Contributions, Publications, and Research Methodology.*

## 1.1 Research Context

Agile Development (AD) has been one of the most important software development paradigms of the Software Engineering community. Agile evangelists argue that agile methodologies bring benefits such as: *(a) faster deliveries*, *(b) enhancement in ability to manage changing priorities*, *(c) improvement in business/IT alignment* *(d) increment in development productivity and quality*, *(e) focus on business value and users*, *(f) reducing project risks and cost*, *(g) enhancement in distributed team management*, *(h) increment in development team and stakeholders engagement*, and *(i) improvement in project visibility* (RAO *et al.* 2011 [3], MEYER 2014 [4], TRIPP and ARMSTRONG 2016 [5], TARWANI and CHUG 2016 [6]).

The promise of such benefits has created a movement towards the adoption of agile methods by the software industry. This movement can be confirmed by studies such as the series *State of Agile$^{TM}$Reports* that, year after year, showed an increment in the number of companies adopting agile development. In the *4th Annual State of Agile$^{TM}$Report 2009* [7], the percentage of respondents that worked in organizations that use agile software development to some degree was 84%. About a decade later, the *14th Annual State of Agile$^{TM}$Report 2020* [8] revealed this percentage increased to 95%.

This interest in agile software development has produced a significant volume of peer-reviewed studies and grey literature (books, blogs, magazines, etc.) that focus on proposing frameworks and guidelines on how agile methodologies can be adopted, combined, and used in software projects (SIDKY *et al.* 2007 [9], QUMER and HENDERSON-SELLERS 2008 [10], ELSSAMADISY 2008 [11], ROHUNEN *et al.* 2010 [12], HAJJDIAB and TALEB 2011 [13], BARLOW *et al.* 2011 [14], GANDOMANI and NAFCHI 2015 [15]).

However, despite the efforts to follow agile practices, mastering agile practices is still a reality for just a few companies that are willing to adopt agile methods. The survey (VERSIONONE 2020 [8]) that pointed out that 95% of the participants worked in companies that somehow adopted agile practices, when analized the agile maturity of these companies, surprisingly revealed that only 5% of them claim they have fully integrated agile practices into their software development process.

As a consequence the following question arises "*what is happening with those organizations (95%) that do not master agile practices?*". Studies such as DE SOUZA BERMEJO *et al.* 2014 [16], ELORANTA *et al.* 2015 [17], LÓPEZ-MARTÍNEZ *et al.* 2016 [18], GREGORY *et al.* 2016 [19], AMBLER 2018 [20] suggest companies may be misusing agile practices without clearly understanding the consequences of the deviations. ELORANTA *et al.* 2015 [17] conducted a survey with 18 teams distributed over 11 organizations to investigate how they are applying the Scrum methodology (ALLIANCE 2016 [21]) and revealed, for example, that: *(a)* when they verified the presence of the Product Owner (which is a role prescribed by Scrum that is responsible for optimizing Return On Investment (ROI)), 7 teams did not have a Product Owner; *(b)* regarding the *Work estimation* (that is an important factor of Scrum to enable an efficient resource allocation and to improve predictability), 2 teams did not do estimates at all and in 6 teams, a project manager or a Product Owner made the estimates; *(c)* when they verified the use of a Product Backlog (which is an artifact prescribed in Scrum to control the project scope), 3 teams were not using it at all; *(d)* when they analyzed the factor *Avoid team interruptions* (that in Scrum means keeping the team focused on the Sprint goal and protecting it from outside interruptions), 13 teams used to be interrupted during the Sprint as team members have to answer phone calls, take bug issues and maintain legacy systems. *(e)* assessing the aspect *Self-organization* (Self-organized teams in Scrum means, among other things, that their members should decide what task from the Sprint backlog should be worked based on a predefined priority), the survey revealed that in 9 cases the teams were not self-organized and relied on the Project Manager to define what should be done and to assign tasks to team members.

Failure to adopt agile practices properly can be harmful and prevent organizations to obtain full benefits of agile methods. The survey with 149 respondents conducted by AMBLER 2018 [20], for example, revealed that 36% of the participants reported that they had experienced challenges in an agile project, and 3% of the participants reported complete failure. DE SOUZA BERMEJO *et al.* 2014 [16] presented a quantitative study that collected data from about 400 companies and concluded that almost a third of them can be characterized as "*organizations that have high rates of agile principles usage and low success rates in software development*". 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 (OZCAN-TOP and DEMIRÖRS 2015 [22]).

To mitigate problems related to the misuse of agile practices, it is important that organizations understand how they are applying agile practices and the improvements opportunities. Some agile methods already prescribe dynamics for process improvement such as the *Sprint Retrospect* in Scrum, the *Reflective Improvement* in Crystal (COCKBURN 2002 [23]), and the *Learning Loop* in ADS (HIGHSMITH 2000 [24]). The *Sprint Retrospect*, for example, is a meeting that usually follows the sprint, where the development team discusses how they conducted the iteration, the main problems they faced and how the process could be improved. These initiatives, although useful as a simple process improvement strategy, do not provide a comprehensive analysis of how an organization is adopting agile practices (PACKLICK 2007 [25]). It is preferable to conduct a more holistic and systematic assessment.

In the context of Software Engineering, Agility Assessment (AA) is an important tool to assist organizations in understanding how they are applying agile practices and their gaps toward an effective agile development (ADALI *et al.* 2016 [26]). There are many types for AA approaches such as agility maturity models, agility checklists, agility surveys, and agility assessment tools (SOUNDARARAJAN and ARTHUR 2011 [27]). In this study, we focus on agility assessment approaches that somehow automate or guide the assessment process.

In the next section, we explore the problems with existing agility assessment approaches that motivated this research.

## 1.2 Problem Statement

Analyzing the state-of-the-art on agility assessment approaches, we have identified the following problems and limitations:

**P1. *Unclear assessment criteria selection***: it is not clear how the assessment criteria were defined in the existing approaches. For example, approaches based on surveys such as STORM-CONSULTING 2008 [28], KREBS 2011 [29], LAGESTEE 2012 [30], INFOTECH 2013 [31], COHN and RUBIN 2015 [32], TOUSIGNANT 2019 [33] do not make it clear how the questions that compose their questionnaires were defined. In STORM-CONSULTING 2008 [28] there is even a section whose title, "*Blue Sky thinking*", does not match any term present in the academic literature on agile development. This problem is relevant because the interpretation of agile principles, values and practices as assessment criteria is a fundamental task in the design of an agility assessment approach TURETKEN *et al.* 2012 [34], LY *et al.* 2015 [35] and should be clearly and explicitly documented. The lack of transparency in the selection of these criteria is a critical threat to the approaches that could be assessing software development aspects not necessarily related to agile development (ADALI *et al.* 2016 [36]).

**P2. *Unclear assessment criteria representation***: this problem refers to the limitations of the approaches in representing the assessment criteria in a explicit and clear manner. Approaches based on spreadsheet KREBS 2011 [29], LAGESTEE 2012 [30], INFOTECH 2013 [31], for example, have their assessment algorithms implemented through undocumented macros and formulae. Other web-based survey approaches STORM-CONSULTING 2008 [28], COHN and RUBIN 2015 [32], MCCALLA and GIFFORD 2016 [37] have a proprietary code, i.e, they do not reveal the algorithms used to process the questionnaire answers. The lack of transparency in representing the assessment criteria jeopardizes the approaches' ability to adapt to different contexts (ADALI *et al.* 2016 [36]). It is risky to rely on static criteria since the assessment context varies from organization to organization and even among projects inside the same organization. For that reason, it is desired that an agility assessment approach enables the specification of assessment criteria using a transparent, expressive, and flexible representation mechanism (MOHA *et al.* 2006 [38], LY *et al.* 2015 [35]).

**P3.** ***Lack of support for adding new assessment criterion***: this problem refers to the lack of support for adding new elements into the set of criteria used by the agility assessment approach. For example, none of the agility assessment approaches based on surveys above mentioned (spreadsheet-based or web-based) KREBS 2011 [29], LAGESTEE 2012 [30], INFOTECH 2013 [31], STORM-CONSULTING 2008 [28], AGILE 2012 [39], COHN and RUBIN 2015 [32] offer an extension mechanism to include new questions into their questionnaires. Similar to the previous problem, this issue restricts the ability of approaches to adapt to different contexts since they only use a predefined set of assessment criteria (MOHA *et al.* 2006 [38], ADALI *et al.* 2016 [36]).

**P4.** ***Manual data collection and input***: this problem is related to the fact that the existing approaches depend mainly on the manual data collection and input to perform an agility assessment. For example, in self-assessment solutions such KREBS 2011 [29], LAGESTEE 2012 [30], INFOTECH 2013 [31], ELIASSEN-GROUP 2013 [40], someone has to enter manually the data into a spreadsheet so it can be analyzed. Manual data collection and input are laborious, slow, costly, error-prone, not scalable (ULLAH *et al.* 2013 [41], DIMYADI and AMOR 2017 [42]), and therefore represent a severe threat to the agility assessment approach.

**P5.** ***Lack of real-time assessment feedback***: this problem refers to the limitation of approaches in not providing real-time assessment. Existing agility assessment solutions usually only perform *post mortem* (COLLIER *et al.* 1996 [43]) analysis, i.e., the assessment is performed after the conclusion of an iteration or a project. Although solutions based on *post-mortem* analysis are useful to investigate how agile practices were applied and which gaps should be addressed in the future, the lack of support for real-time analysis limits the approaches' ability to identify the misuse of agile practices during software development and to prevent unwanted outcomes (PARK *et al.* 2012 [44], LY *et al.* 2015 [35]).

**P6.** ***Limited Scalability***: this problem refers to the limitation of approaches to scale, i.e., to be able to perform the assessment at the same level of functionality and usability, regardless of the size of the project or organization. This issue is caused, in part, by the limitations pointed in P4 (*Manual data collection and input*) and P5 (*Lack of real-time assessment feedback*). For example, the existing approaches above mentioned, as well as other approaches discussed in Chapter 2, require someone to

Table 1.1: Code smell x agile smell metaphors

| | |
|---|---|
| *A **code smell*** | *An **agile smell*** |
| *describes the identification* | *describes the identification* |
| *of early warnings signals* | *of early warnings signal*s |
| *that something in* | *that something in* |
| ***computer code*** | ***an agile project/organization*** |
| *may need to be* | *may need to be* |
| ***rewritten**[45].* | ***adjusted**.* |

manually collect and enter data into the solutions. This interference of human users may take a significant amount of time and effort to perform, making it challenging to use these approaches on a large scale.

## 1.3 Research Goals

In this study, we aim to contribute to the area of agility assessment by proposing an agility assessment approach that mitigates the problems presented in the previous section. The proposed approach is based on the "***agile smells***" concept.

We are adapting the *code smell* metaphor (FOWLER *et al.* 1999 [45]) to the context of agility assessment and introducing the *agile smell* metaphor as illustrated in Table 1.1. According to Fowler and Beck, who popularized the term in (FOWLER *et al.* 1999 [45]), a *code smell* denotes an indication that may correspond to a deeper problem in the software source code or architecture that need to be fixed or refactored. We are using the *agile smell* term to denote a practice that may impair the proper adoption of an agile practice. Then, the examination of early warning signals may indicate that certain aspects of agile in a project or organization need to be adjusted.

The main goal of this study is to **propose an agility assessment approach that mitigates the problems P1 to P6 (described in Section 1.2), while providing a solid foundation for defining an infrastructure to support Agility Assessment**. To achieve the main goal, we defined 4 fine-grained goals:

**G1.** ***Define a catalogue of agile smells***: define, through a methodological process, a catalogue of agile smells that guides the agility assessment approach and makes explicit the relation between the assessment criteria and the agile practices that motivated them. This goal aids to address the problem **P1** (*Unclear* *assessment criteria selection*).

**G2.** ***Define an agile smell representation approach***: define an expressive approach to represent the agile smells. The approach must be expressive and flexible to enable the specification of the agile smells presented in the catalogue produced by the goal **G1** (*Define a catalogue of agile smells*) as well as support the addition of new agile smells. This goal aids to address the problems **P2** (*Unclear assessment criteria representation*) and **P3** (*Lack of support for adding new assessment criterion*).

**G3.** ***Define a data extraction approach***: define a data extraction mechanism capable of loading data from different sources and using it as input for the agility assessment approach. This goal aids to address the problems **P4** (*Manual data collection and input*), **P5** (*Lack of real-time assessment feedback*), and **P6** (*Limited Scalability*).

**G4.** ***Define and implement a computational system***: define and implement a computational system that supports the specification and detection of agile smells. This goal supports the goals **G2** (*Define an agile smell representation approach*) and **G3** (*Define a data extraction approach*) and aids to address the problems **P5** (*Lack of real-time assessment feedback*) and **P6** (*Limited Scalability*).

## 1.4 Research Contributions

The main contributions of this research can be summarized as follows:

**C1.** ***AgileQube Approach***: an agility assessment approach that defines the steps and the elements necessary to automatically (or semi-automatically) identify agile smells in agile projects.

**C2.** ***Catalogue of Agile Smells***: a set of agile smells that were identified through a literature review, confirmed by a survey and organized in a structured format.

**C3.** ***Agile Project Metamodel***: a metamodel to support the collection of data from agile projects.

**C4.** ***Agile Smell Schema***: a structure schema to support the representation of agile smells.

**C5.** ***AgileQube App***: a computational system to support the execution of some phases from the *AgileQube Approach*.

## 1.5 Publications

So far, this research has produced the following peer-reviewed publications:

1. ***BPMN-R: An extension to BPMN*** (*exploratory proposal*)
   TELEMACO, U., OLIVEIRA, T. BPMN-R: An extension to BPMN to Represent Software Process Rules: doctoral symposium. In: *XV Workshop de Teses e Dissertações em Qualidade de Software (WTDQS 2017)*, 2017 [46]

2. ***Catalogue of Agile Smells*** (*preliminary version*)
   TELEMACO, U., OLIVEIRA, T., ALENCAR, P., et al. A catalog of bad agile smells for agility assessment. In: *Proceedings of the 2019 Ibero-American Conference on Software Engineering*, CIbSE 2019, pp. 30–43, 2019 [47];

3. ***Agile Project Metamodel***
   TELEMACO, U., OLIVEIRA, T., ALENCAR, P., et al. A metamodel for representing agile software development projects. In: *Proceedings of the 2019 Ibero-American Conference on Software Engineering*, CIbSE 2019, 2019 [48];

4. ***Catalogue of Agile Smells*** (*extended version*)
   TELEMACO, U., OLIVEIRA, T., ALENCAR, P., et al. A Catalogue of Agile Smells for Agility Assessment, *IEEE Access*, v. 8, pp. 79239–79259, 2020 [49].

A manuscript describing the *AgileQube Approach* and its main components is almost ready and will be soon submitted to a specialized journal.

We have also published, as open-source projects, two modules that support the computational system:

1. ***Prisma KIP Domain*** [1]: a reference implementation of the *Agile Project Metamodel*;

2. ***Prisma ZenHub ETL*** [2]: an ETL module to extract data from the *ZenHub* platform.

[1]`https://github.com/utelemaco/prisma-kip-domain`
[2]`https://github.com/utelemaco/prisma-zenhub-etl`

## 1.6 Research Methodology

To accomplish the goals and contributions, this research was organized in four stages based on the methodology proposed by PEFFERS *et al.* 2007 [1]. An overview of these stages and the effort distribution along the PhD journey is presented in Figure 1.1. The macro-activities performed in this research are represented in Figure 1.2.

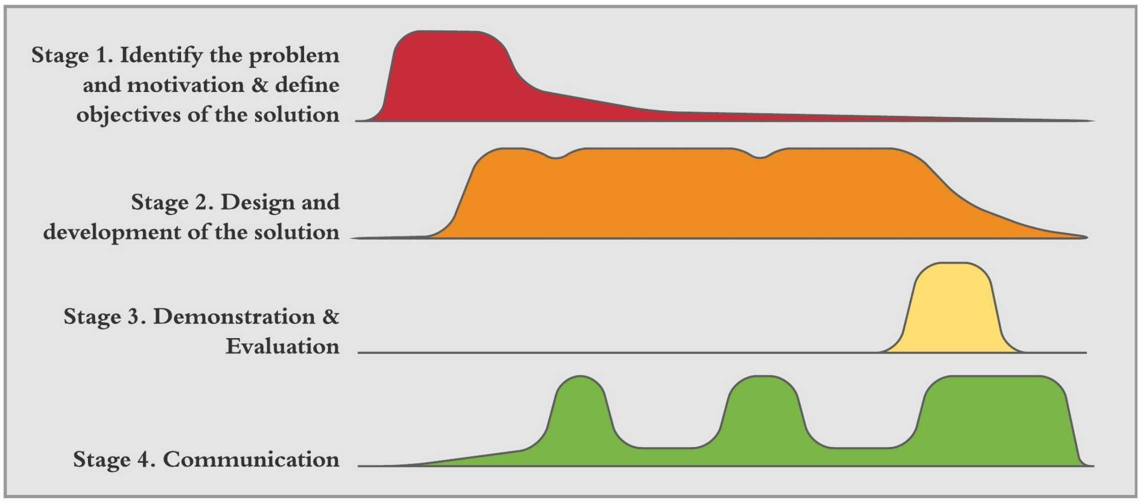


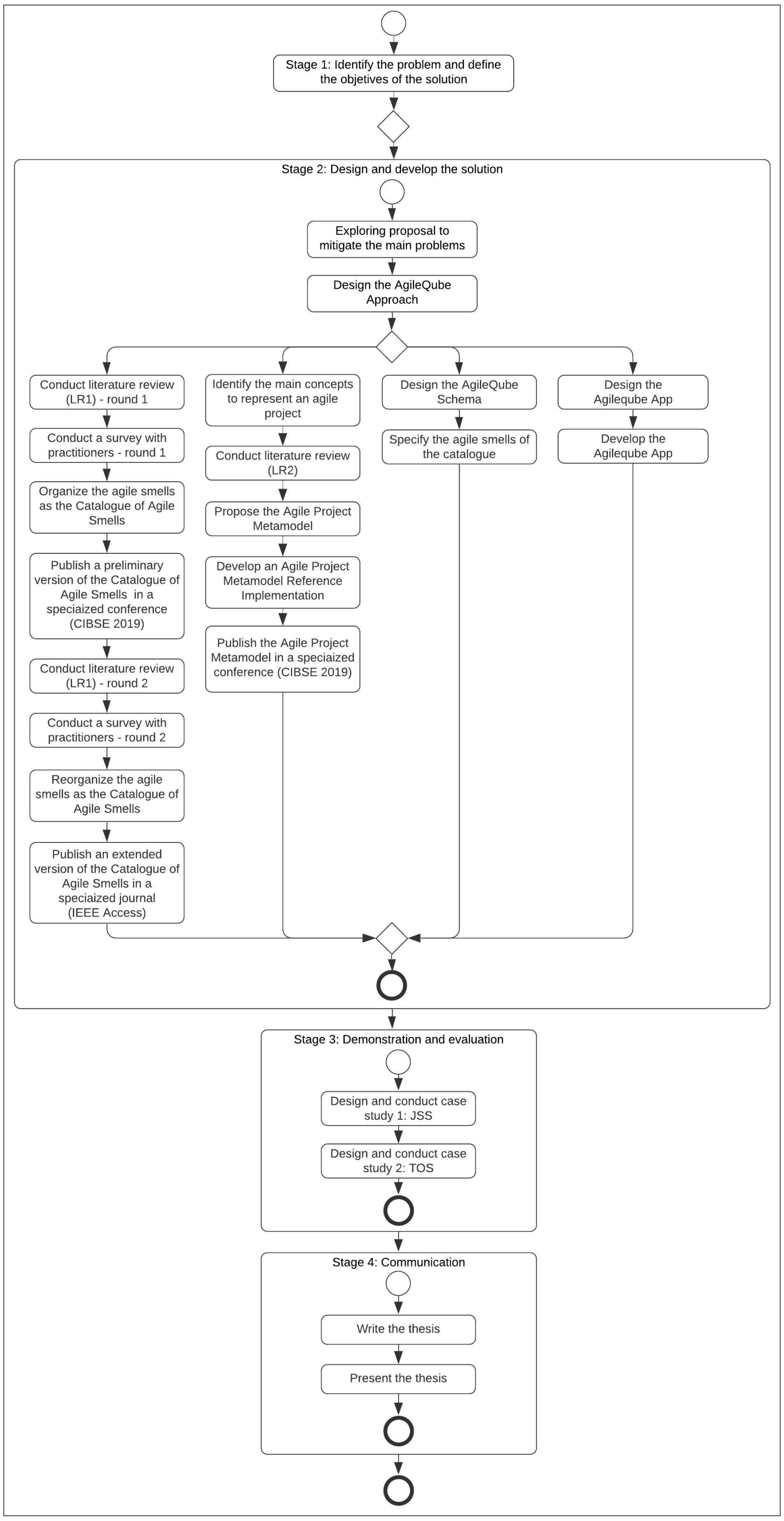
Figure 1.1: Stages of the research

**Stage 1. Identify the problem and define the objectives of the solution**

At the very early stage of this research, we conducted an *ad hoc* literature review to confirm the importance of agility assessment to the agile community, learn about existing approaches and identify the main problems and limitations of these approaches. At the end of this phase, we have identified a preliminary lists of problems and goals. These lists have been evolved during the research and the final result is presented in Sections 1.2 and 1.3.

**Stage 2. Design and development of the solution**

We started this stage by working on exploring proposals to mitigate the problems identified in the first stage. We tried some ideas such as the use of approaches based on *process compliance checking and monitoring* (LU *et al.* 2008 [50], LY *et al.* 2015 [35], DE MELLO *et al.* 2016 [51]). However, these approaches did not seem reasonable in such a flexible and dynamic environment as the agile software development environment. Then, we decided to adapt the abstraction of *code smell* to the context of agile development and introduced the *agile smell* term. We designed a solution to automatically detect agile smells (*AgileQube Approach*) and identified other 4



Stage 1: Identify the problem and define the objetives of the solution
Stage 2: Design and develop the solution
Exploring proposal to mitigate the main problems
Design the AgileQube Approach
Conduct literature review (LR1) - round 1
Conduct a survey with practitioners - round 1
Organize the agile smells as the Catalogue of Agile Smells
Publish a preliminary version of the Catalogue of Agile Smells in a speciaized conference (CIBSE 2019)
Conduct literature review (LR1) - round 2
Conduct a survey with practitioners - round 2
Reorganize the agile smells as the Catalogue of Agile Smells
Publish an extended version of the Catalogue of Agile Smells in a speciaized journal (IEEE Access)
Identify the main concepts to represent an agile project
Conduct literature review (LR2)
Propose the Agile Project Metamodel
Develop an Agile Project Metamodel Reference Implementation
Publish the Agile Project Metamodel in a speciaized conference (CIBSE 2019)
Design the AgileQube Schema
Specify the agile smells of the catalogue
Design the Agilequbе App
Develop the Agilequbе App
Stage 3: Demonstration and evaluation
Design and conduct case study 1: JSS
Design and conduct case study 2: TOS
Stage 4: Communication
Write the thesis
Present the thesis

Figure 1.2: Macro-activities performed in the research in the BPMN notation (OMG 2011 [2])

additional contributions: *Catalogue of Agile Smells*, *Agile Project Metamodel*, *Agile Smell Schema*, and *AgileQube App*. The remainder of Stage 2 was conducted in 4 *threads* (one for each additional contribution).

A preliminary version of the *Catalogue of Agile Smells* was elaborated through three steps: *(1)* we conducted a literature review (LR1) to identify a set of agile smells; *(2)* we conducted a survey with practitioners to confirm the agile smells; and *(3)* we organized the agile smells as a catalogue. After submitting the preliminary results to a conference and receiving feedback from a specialized community, we produced a final version of the *Catalogue of Agile Smells* through the following steps: *(1)* we updated the literature review (LR1); *(2)* we conducted the survey with more participants; and *(3)* we reorganized the agile smells in the catalogue according to new results.

The *Agile Project Metamodel* was elaborated in four steps: *(1)* we identified a set of required concepts; *(2)* we conducted a literature review (LR2) to investigate how the existing software development metamodels can represent these concepts; *(3)* we extended the existing metamodels and proposed the *Agile Project Metamodel*; and *(4)* we developed a reference implementation of the proposed metamodel.

In Stage 2, we also designed and developed the *Prisma ZenHub ETL*, the *Agile Smell Schema*, and the *AgileQube App*.

**Stage 3. Demonstration and Evaluation**

In Stage 3, we designed and conducted two case studies (*Journal Submission System* and *Terminal Operational System*) according to RUNESON and HÖST 2009 [52] guidelines to support the demonstration and evaluation of the proposed approach.

**Stage 4. Communication**

We published, in a specialized conference, a preliminary version of the *Catalogue of Agile Smells* (TELEMACO *et al.* 2019 [47]) and the *Agile Project Metamodel* (TELEMACO *et al.* 2019 [48]). An extended version of the catalogue was published in the IEEE Access journal (TELEMACO *et al.* 2020 [49]). We published as open source projects on *GitHub* a reference implementation of the *Agile Project Metamodel* (*Prisma KIP Domain*[3]) and an ETL module for the *ZenHub* platform (*Prisma ZenHub ETL*[4]).

[3]https://github.com/utelemaco/prisma-kip-domain
[4]https://github.com/utelemaco/prisma-zenhub-etl

We are finishing a manuscript describing the *AgileQube Approach* and its main components that will be soon submitted to a specialized journal.

The writing and presentation of this thesis conclude this stage.

## 1.7 Outline

The remainder of this thesis is organized in the following structure: Chapter 2 presents the theoretical foundation of this research and discusses the related work. The *AgileQube Approach* is presented in Chapter 3. Chapters 4, 5, and 6 presents, respectively, the *Catalogue of Agile Smells*, the *Agile Project Metamodel*, and the *Agile Smell Schema*. The computational infrastructure (*AgileQube App*) that supports the proposed approach is presented in Chapter 7. Chapter 8 discusses two case studies (*Journal Submission System* and *Terminal Operational System*) that were conducted to validate the proposed approach. Chapter 9 presents the final remarks and concludes this thesis.

# Chapter 2

# Background

*This chapter discusses some topics that compose the background of this research.*

## 2.1 Agile Development

In 2001, as a response to a community that demanded more flexible processes, a group of 17 software development professionals met to discuss alternative software development methodologies. The demand for such a innovative approach was mostly justified by the lack of flexibility in traditional approaches such as the Waterfall Model that was dominant at that time. HIGHSMITH 2000 [24] pointed that the cost of change (that grows through the software's development life cycle) at an advanced phase of the project was one of the main reasons for project failure at the end of the 90's. Having a clear vision of the flexible, lightweight and team-oriented software development approach, the 17 practitioners proposed the *Manifesto for Agile Software Development* (BECK *et al.* 2001 [53]) that summarized the fundamental principles of the new approach:

*We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
***Individuals and interactions*** *over processes and tools*
***Working software*** *over comprehensive documentation*
***Customer collaboration*** *over contract negotiation*
***Responding to change*** *over following a plan*

Complemented with the *Twelve Principles of Agile Software*, the manifesto has influenced the software development community and inspired many agile methodologies. But the manifesto was not a pioneer in proposing an agile approach for software development. Actually the history of agile approaches goes way back to 1957, when John von Neumann, Geral Weinberg, Bernie Dimsdale, and Herb Jacobs were applying incremental development techniques for software that were built in IBM and Motorola. Although not knowing how to classify the approach, they were practicing, they realized clearly that it was different from the Waterfall Model in many ways (LARMAN and BASILI 2003 [54]). The main contribution of the manifesto was packing a set of existing values and principles and creating a philosophy that has come to be a universal and efficient new way to manage software projects.

### How agile works

In traditional waterfall methods, the development approach follows a strict linear order that typically involves the stages: *requirement*, *design*, *implementation*, *verification*, and *maintenance*. The business requirements and the project scope are defined at the beginning of the project and the subsequent phases are planned to accommodate these requirements and scope. At the end of the project, the software is delivered to the customers and stakeholders. The agile methodologies, on the other hand, take an iterative approach and software development are conducted through a number of smaller cycles (iterations or sprints). Each cycle is a process development instance in miniature: it has a backlog and a set of orchestrated tasks organized in stages such as: requirement, design, coding, testing, and deployment.

At the end of each development cycle, a potentially shippable piece of software is delivered. Thus, with every iteration new features are added to the product, which results in the gradual project growth. With the features being validated early in the development life cycle, the chances of delivering a potentially failed product are lower.

### Agile Frameworks

There is no formal agreement on the meaning of the concept of “agile”. However, it is common sense to denote as “agile” those software development processes or methods that are shaped around the values and principles proposed in the *Manifesto for Agile Development*. These methods include, but are not limited to: *XP* (BECK 1999 [55], BECK 2000 [56], CUNNINGHAM 1999 [57]), *Scrum* (SCHWABER 1997 [58], SCHWABER and BEEDLE 2001 [59], ALLIANCE 2016 [21], BERTEIG 2015 [60]), *Crystal Family* (COCKBURN 2002 [23]), *Feature*

*Driven Development (FDD)* (PALMER and FELSING 2001 [61], LUCA 1999 [62]), *Dynamic Systems Development Method* (STAPLETON 1997 [63]), *Adaptive Software Development* (HIGHSMITH 2000 [24]), and *OpenUp* (ABRAHAMSSON *et al.* 2002 [64]). Extreme Programming (XP) and Scrum are the most popular of these agile methodologies.

### XP

In (BECK 1999 [55]), Beck introduced XP as "*an approach that turns the conventional software process sideways and rather than planning, analyzing, and designing for the far-flung future, XP exploits the reduction in the cost of changing software to do all of these activities a little at a time, throughout software development*". That revolutionary approach Beck was presenting was organized around 12 interconnected set of software development practices (aka XP practices):

1. *40-hour week*
2. *Coding Standard*
3. *Collective Ownership*
4. *Continuous Integrat.*
5. *Metaphor*
6. *Onsite Customer*
7. *Pair Programming*
8. *Planning Game*
9. *Refactoring*
10. *Simple Design*
11. *Small Releases*
12. *Testing*

### Scrum

First described by TAKEUCHI and NONAKA 1986 [65] as an adaptive, quick, self-organizing product development method and later adapted by SCHWABER and BEEDLE 2001 [59] to the context of software development, Scrum is the most adopted agile framework. It is used exclusively by 58% of organizations while another 27% of the companies combine it with other frameworks (VERSIONONE 2020 [8]). It differs from XP mainly by its nature. While XP is essentially organized around its 12 practices, Scrum is more prescriptive (or process-like) and its structure is based on a set of phases (*pre-game*, *development*, and *post-game*), roles (*Scrum Master*, *Product Owner*, and the *Scrum Team*), artifacts (*Product backlog*, *Sprint backlog*, and *Sprint burndown chart*) and activities (*Daily Scrum*, *Sprint Planning*, *Sprint Review*, and *Sprint Retrospective*). An Scrum team should consist of up to 7 team members, in order to stay flexible and productive. A basic unit of work in scrum - sprint - is a short development cycle (between 1 and 4 weeks long) that is needed to produce a shippable product increment.

### Agile Practices

Although there is no conventional set of “agile practices” (in fact, the *Manifesto for Agile Development* focus more on values and principles rather than practices), some studies tried to identified engineering practices and techniques usually associated with the concepts of agility. ABRANTES and TRAVASSOS 2011 [66] conducted a study to identify the most commonly used agile practices that revealed the following set:

1. *Coding Standards*
2. *Collective Code Ownership*
3. *Continuous Integration*
4. *Metaphor*
5. *Onsite Customer*
6. *Open workspace*
7. *Pair Programming*
8. *Planning Game*
9. *Product Backlog*
10. *Project Visibility*
11. *Refactoring*
12. *Simple Design*
13. *Small Releases*
14. *Stand-up meetings*
15. *Sustainable Pace*
16. *Test-Driven Development*
17. *Whole Team*

A recent survey (VERSIONONE 2020 [8]) conducted in the software industry listed, as the most employed agile techniques, the following items (ordered by percentage of use):

1. *Daily Standup (85%)*
2. *Sprint/Iteration Retrospectives (81%)*
3. *Sprint/Iteration Planning (79%)*
4. *Sprint/Iteration Review (77%)*
5. *Short Sprint/Iteration (64%)*

This survey also listed, as the most adopted engineering practices associated to agility, the following items (ordered by percentage of use):

1. *Unit Testing (67%)*
2. *Coding Standards (58%)*
3. *Continuous Integration (55%)*
4. *Refactoring (43%)*
5. *Continuous Delivery (41%)*
6. *Automated Acceptance Testing (36%)*

7. *Continuous Deployment (36%)*
8. *Pair Programming (31%)*
9. *Test-Driven Development (30%)*
10. *Collective Code Ownership (29%)*
11. *Sustainable Pace (23%)*
12. *Behavior-Driven Development (19%)*
13. *Emergent Design (13%)*

## 2.2 Agility Assessment

Although many companies pursue agility on their software development processes, the proper adoption of so-called agile practices is not straightforward and some authors refer to it as "agile journey" or "agile transformation" (QUMER and HENDERSON-SELLERS [10], FRASER *et al.* [67], GANDOMANI *et al.* [68, 69], SAHOTA [70]). The *14th Annual State of Agile Survey*$^{TM}$ (VERSIONONE 2020 [8]) confirmed this by revealing that, although 95% of companies surveyed claimed they are using agile practices, only 5% of the companies indicated they achieved the high level of competency with agile practices. In such a challenging scenario, it is important for organizations to identify their gaps in agile practices, otherwise, they may not receive the benefits of adopting an agility approach (AMBLER and LINES 2012 [71]).

Agility Assessment (AA) is an important tool to assist organizations in understanding how they are applying agile practices and their gaps towards an effective agile development (ADALI *et al.* 2016 [26]). AA encompasses 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 assessment checklists, agility assessment surveys, and agility assessment tools (SOUNDARARAJAN and ARTHUR 2011 [27]) and they can be organized in the following categories: *(a) text-based*; *(b) game-based*; *(c) spreadsheet-based*; *(d) graph-based*; and *(e) web-based* (ADALI *et al.* 2016 [36]).

*Text-based* approaches include guidelines, surveys, or checklist available as plain documents. They are mainly executed without the support of a specialized computational infrastructure. Although it is possible to use a generic software such as a text editor or a spreadsheet to support some phases of the evaluation process. *Agile Journey Index* (KREBS 2011 [29]), *A Corporate Agile 10-point Checklist* (YATZECK 2012 [72]), and *Comparativity Agility* (WILLIAMS *et al.* 2010 [73]) are examples of *text-based* agility assessment approaches.

*Game-based* approaches are similar to *text-based* solutions in many ways. They are supported by plain documents (such as game instruction or card deck) and there is none or few specialized tools. But they differ from *text-based* approaches by proposing assessment strategies based on games. *Retropoly* (SFIRLOGEA and GEORGESCU 2017 [74]), *The Agile Self-assessment Game* (LINDERS 2019 [75]), and *Team Barometer* (JANLÉN 2014 [76]) are some examples of *game-based* approaches.

*Spreadsheet-based* approaches are supported by pre-configured spreadsheets where someone manually enters data on input cells and the results of the agility assessment are calculated by specific formulae and shown in output cells. Input cells can be set up with predefined options and outputs cells can be color-coded to enhance user-friendliness. As examples of *spreadsheet-based* approaches, we can mention *Agile Health Dashboard* (LAGESTEE 2012 [30]), *Agility Questionnaire* (BRITSCH 2017 [77]), and *Team and Technical Agility Self-Assessment.*

*Graph-based* approaches are similar to *Spreadsheet-based* tools. They are typically supported by pre-configured spreadsheets where someone inputs data on specific cells. But they differ from *spreadsheet-based* solutions by displaying the assessment results using graphs. *Depth of Kanban* (ACHOUIANTZ 2013 [78]), and *Lean/Agile Depth Assessment Checklist A3* (YERET 2013 [79]) are examples of *graph-based* tools.

*Web-based* approaches are those supported by specialized web-based applications (aka web apps). Solutions on this category include online surveys and questionnaires where someone enters data on input fields and the system calculates the agility assessment results. As examples of *web-based* approaches, we can mention *Agile Maturity Assessment* (TOUSIGNANT 2019 [33]), *Measure.Team* (ALBRECHT and EDDINGS 2020 [80]), and *Scrum Checklist* (KNIBERG 2012 [81]).

In the next section, we present a comprehensive analysis of the main agility assessment approaches and tools related to this research.

## 2.3 Agility Assessment Approaches and Tools

This section focuses on presenting studies that proposed approaches or tools that aim at supporting agility assessment and discuss how they differ from the approach proposed in this research.

After a careful review of relevant studies in journals and proceeding, as well as by using the Google search engine, we identified 60 approaches and tools related to this research. The selected approaches and tools are alphabetically listed in Table 2.1. Appendix A presents a description of these approaches.

Table 2.1: Approaches and tools related to this research

| # | Name | Type |
|---|---|---|
| 1 | 42 point test: How Agile are You [82] | text-based |
| 2 | A Better Team [83] | text-based |
| 3 | A Corporate Agile 10-point Checklist [72] | text-based |
| 4 | Agile Adoption Interview [84] | web-based |
| 5 | Agile Alert [85] | web-based |
| 6 | Agile Assessment [86] | spreadsheet |
| 7 | Agile Enterprise Survey [87] | web-based |
| 8 | Agile Excellerate [88] | web-based |
| 9 | Agile Health Dashboard [30] | spreadsheet |
| 10 | Agile Journey Index (AJI) [29] | text-based |
| 11 | Agile Maturity Assessment [33] | web-based |
| 12 | Agile Skills Self-Assessment [89] | web-based |
| 13 | Agile Team Evaluation [90] | text-based |
| 14 | Agility Maturity Self Assessment [91] | text-based |
| 15 | Agility Maturity Self Assessment Survey [92] | text-based |
| 16 | Agility Questionnaire [77] | spreadsheet |
| 17 | Back-of-a-Napkin Agile Assessment [93] | text-based |
| 18 | Balbes' Agility Assessment [94] | text-based |
| 19 | Borland Agile Assessment [95] | text-based |
| 20 | Business Agility Manifesto [96] | text-based |
| 21 | Cargo Cult Agile Checklist [97] | text-based |
| 22 | Comparative Agility$^{TM}$(CA) [73] | text-based |
| 23 | Comprehensive Agility Measurement Tool (CAMT) [98] | text-based |
| 24 | Depth of Kanban [78] | graph-based |
| 25 | DIB Guide [99] | text-based |
| 26 | Enterprise and Team Level Agility Maturity Matrix [40] | spreadsheet |
| 27 | Enterprise Business Agility Maturity Survey [100] | text-based |
| 28 | Five Key Numbers to Assess Agile Engineering Practices [101] | text-based |

**Continued on next page**

| Continuation of Table 2.1 | | |
|---|---|---|
| # | Name | Type |
| 29 | GAO's Agile Assessment Guide [102] | text-based |
| 30 | How Agile are you? A 50 Point Test [103] | web-based |
| 31 | IBM DevOps Practices Self-Assessment [104] | text-based |
| 32 | Joe's Unofficial Scrum Checklist [105] | text-based |
| 33 | Karlskrona Test [106] | text-based |
| 34 | Kanban Maturity Assessment [107] | web-based |
| 35 | Lean Agile Intelligence [37] | web-based |
| 36 | Lean/Agile Depth Assessment Checklist A3 [79] | graph-based |
| 37 | Lebow's Agile Assessment [108] | text-based |
| 38 | Measure.Team [80] | web-based |
| 39 | Nokia Test [109] | text-based |
| 40 | Objectives Principles Strategies (OPS) [110] | text-based |
| 41 | Open Assessments [111] | web-based |
| 42 | Organizational Agility Self-Assessment [112] | spreadsheet |
| 43 | People 10 Team Assessment Approach [113] | text-based |
| 44 | Perceptive Agile Measurement (PAM) [114] | text-based |
| 45 | Quick Self-Assessment of Your Organization's Agility [115] | text-based |
| 46 | Retropoly [74] | game-based |
| 47 | Scrum Assessment Series [116] | text-based |
| 48 | Scrum Checklist [81] | web-based |
| 49 | ScrumMaster Checklist [117] | text-based |
| 50 | Self Assessment Tool for Transitioning to Agile [118] | text-based |
| 51 | Squad Health Check Model [119] | game-based |
| 52 | Team and Technical Agility Self-Assessment [39] | spreadsheet |
| 53 | Team Barometer [76] | game-based |
| 54 | TeamMetrics [120] | web-based |
| 55 | Test Maturity Card Game [121] | game-based |
| 56 | The Agile Self-Assessment Game [75] | game-based |
| 57 | The Art of Agile Development [122] | text-based |
| 58 | The Joel Test: 12 Steps to Better Code [123] | text-based |
| 59 | Visual Management Self-Assessment [124] | web-based |
| Continued on next page | | |

| Continuation of Table 2.1 | | |
|---|---|---|
| # | **Name** | **Type** |
| 60 | Yodiz's Team Agility Self Assessment [125] | spreadsheet |
| **End of Table 2.1** | | |

One of the main limitations of the presented approaches is related to their dependency on manual execution. Indeed, all analyzed agility assessment approaches are executed almost completely manually. For example, the text-based and game-based approaches are carried out entirely manually. The execution of these approaches may involve the following manual tasks: (a) assessment preparation; (b) data collection; (c) data analysis; and (d) interpretation of data to decide whether the evaluation criteria were met. For spreadsheet-based, graph-based, and web-based approaches, the situation is not so different and someone willing to use these approaches still has to spend a substantial effort executing manual tasks such as: (a) assessment preparation; (b) data collection; and (c) data analysis. The problem with agility assessment approaches that rely heavily on manual execution is that they tend to be more error-prone, costly, and time-consuming. Besides being difficult to scale and to provide real-time feedback.

The lack of opportunities for automation is another limitation of these approaches. In fact, the approaches were not designed with requirements for automation in mind. For approaches based on questionnaires, one factor that contributes to this limitation is the nature of some questions. For example, how to automate the data collection and interpretation for questions such as "*Is the team empowered to make decisions?*" or "*Does the team have a clear vision?*"? Other approaches proposed assessment criteria that are equally hard to be automated and hence require someone to manually assess them.

Another limitation presents in almost all the presented approaches came from the fact the authors failed to make clear the relation between the assessment criteria and the agile principles, values, and practices. For example, the approaches based on questionnaires do not explain how the questions were elaborated and how they relate to agile principles, values, and practices. The spreadsheet-based and web-based approaches do not present any technical documentation about the macros and algorithms used to calculate the assessment results.

A limitation observed in the spreadsheet-based and web-based approaches is the lack of an extension mechanism to enable the tailoring of the assessment criteria (i.e., the addition of new criteria, and the configuration or removal of existing criteria).

Regarding the efficiency and efficacy of the analyzed agility assessment approaches, as most of the proposals are not academically created, their empirical validity is not confirmed.

## 2.4 Agility Maturity Models

In this section we present studies that proposed agility maturity models and discuss how they relate to this research.

NAWROCKI *et al.* 2001 [126] proposed the *XPMM*, a 4-level maturity model for *eXtreme Programming* (XP) based on CMMI v1.02 (TEAM 2000 [127]). The authors mapped XP practices to CMM Key Practice Areas (KPAs) and defined the following levels: *Level 1. Not compliant at all*; *Level 2. Initial*; *Level 3. Advanced*; and *Level 4. Mature*. Each level has a specific set of obligatory practices and to be classified at a given level, an organization has to follow all the practices assigned to that level and all the practices of the lower levels. Using a similar approach, LUI and CHAN 2005 [128] proposed a 4-stage road map for implementing *eXtreme Programming* in a software team. Each maturity stage has a set of XP practices defined by their interrelations.

The *Agile Maturity Map* (PACKLICK 2007 [25]) is an agile maturity model organized in 5 levels: *Level 1. Awareness*, *Level 2. Transformation*, *Level 3. Breakthrough*, *Level 4. Optimizing*, and *Level 5. Mentoring*. Different from other agile maturity models that organize their levels in terms of agile practices, the levels in the Agile Maturity Map are organized in terms of goals. For example, the level *Awareness* is achieved when: "*The team understands the goals, and understands the value of pursuing the goals and their acceptance criteria. Awareness of existing 'better' practices around the goals typically exists as well. The team may possibly implement basic activities to address the goal*".

PATEL and RAMACHANDRAN 2009 [129] proposed the *Agile Maturity Model* that organizes the agility maturity in 5 levels: *Level 1. Initial*, *Level 2. Explored*, *Level 3. Defined*, *Level 4. Improved*, and *Level 5. Sustained*. Each level is composed of a specific set of key process areas (KPA) (e.g., *project planning*, *on-site customer availability*) and questionnaires that enable the assessment of the current state of an organization or project and the identification of the KPAs that need improvement.

OZCAN-TOP and DEMIRÖRS 2015 [22] developed the *AgilityMod*, a comprehensive software agility assessment reference model based on the structure of the standard ISO/IEC 15504 (ISO/IEC 15504 [130]). The *AgilityMod* has 2 dimensions: aspect dimension and agility dimension. The aspect dimension defines 4 aspects (*Exploration*, *Construction*, *Transition*, and *Management*) which correspond to maturity stages. The agility dimension defines 4 agility levels (*Level 0. Not Implemented*, *Level 1. Ad-Hoc*, *Level 2. Lean*, and *Level 3. Effective*) which represent the agility capability of the aspects. Each aspect has a set of agile practices and might be at one of the 4 levels. When an aspect's agility progresses, its conformance to agile values and principles increases.

*SAFe Maturity Model* (SAFe MM) (TURETKEN *et al.* 2017 [131]) is a maturity model that aids organizations in defining a roadmap for adopting SAFe (LEFFINGWELL 2016 [132]). The maturity model is organized in 5 levels, 5 principles and 62 SAFe practices that together can also be used to assess the level of SAFe adoption.

Other agility maturity models include PETTIT 2006 [133] and AMBLER 2009 [134].

In general, these models are elaborated using the following structure: they define their agility maturity levels and provide the basic characteristics of each level which include an assessment strategy (i.e., a mechanism to verify if an organization achieved a given level). In a model whose levels are oriented by agile practices, the assessment approach consists of verifying whether the organization is performing the agile practices assigned to the given level. Other models prescribe their specific assessment approaches that could involve a self-assessment survey or a questionnaire that is filled by the Project Manager.

These models share some of the limitations present in the approaches discussed in the previous section including their dependency on manual execution and the lack of opportunities for automation. A factor that contributes to these limitations is that, in all analyzed models, the assessment criteria are hard to be automated and hence require someone to manually verify them.

## 2.5 Agile Adoption Framework

In this section we present studies that proposed approaches to assist organization introducing agile practices into their development processes. These studies are related to this research since they define assessment strategies to verify whether the organization willing to adoption agile practices are applying these practices properly.

In (QUMER and HENDERSON-SELLERS 2008 [10]), the authors presented two contributions: *Agile Software Solution Framework* (ASSF) and *Agile Adoption and*

*Improvement Model* (AAIM). While ASSF is a framework to aid managers assessing the degree of agility they require and how to identify appropriate ways to introduce this agility into their organization, AAIM aims to indicate the current degree of agility of an organization. AAIM categorizes agility in 6 agile levels embedded in 3 agile blocks as follows: (1) Block *Prompt*: *AAIML 1 Agile infancy*; (2) Block *Crux*:*AAIML 2 Agile Initial*, *AAIML 3 Agile Realization*, and *AAIML 4 Agile Value*; and (3) Block *Apex*: *AAIML 5 Agile Smart*, and *AAIML 6 Agile Progress.* The degree of agility of an organization is measured quantitatively by using the 4-DAT tool (QUMER and HENDERSON-SELLERS 2006 [135]) (an agility measurement modelling approach). In 4-DAT, agility is measured in terms of the following features: *Flexibility* (*FY*), *Speed* (*SD*), *Leanness* (*LS*), *Learning* (*LG*) and *Responsiveness* (*RS*). To define the value of these features, that may be 0 or 1, the project manager has to answer a questionnaire composed of a yes/no question per feature. For example, the question that corresponds to the feature *Flexibility* is "*Does the method accommodate expected or unexpected changes?*". The limitations of this approach include: (a) the lack of objective criteria to assess agility and, hence, the result of this approach may be threatened by the bias of the person that is responding the questions; (b) the difficulty of implementing an automated platform to support the agility assessment.

SIDKY *et al.* 2007 [9] presented an agile adoption framework composed of two components: (a) an agile measurement tool, named *Sidky Agile Measurement Index* (SAMI) and (b) a four-stage process, that together guide and assist the agile adoption efforts of organizations. SAMI encompasses five agile levels that are used to identify the agile potential of projects and organizations. The four-stage process, on the other hand, helps determine (a) whether or not organizations are ready for agile adoption, and (b) guided by their potential, what set of agile practices can and should be introduced.

SURESHCHANDRA and SHRINIVASAVADHANI 2008 [136] presented an agile adoption framework for distributed projects composed of 4 stages: *Evaluation*, *Inception*, *Transition*, and *Steady State.* To support the first stage, *Evaluation*, the authors developed a tool similar to SAMI that aims at evaluating the degree of agility and formal ceremonies needed in a given project. The tool assesses factors such as (a) the benefits achievable by following agile, (b) the need for formal communications, (c) the need for extensive training, and (d) the extent of documentation to determine the extent of agility realizable for a project. These studies differ from ours because the presented tools aim to identify the agile potential (i.e, the degree to which a project or an organization can adopt agile practices) instead of assessing the current agility capacity of a project or an organization.

Other approaches focused on proposing approaches to support the adoption of agile development include ELSSAMADISY 2008 [11], ROHUNEN *et al.* 2010 [12], BARLOW *et al.* 2011 [14], HAJJDIAB and TALEB 2011 [13], ESFAHANI 2012 [137], GANDOMANI and NAFCHI 2015 [15], SAHOTA 2012 [70], GLOVER 2012 [138].

As observed in the studies presented in the previous sections, the analyzed frameworks for agile adoption also have as main limitations the dependence on manual execution and the lack of automation opportunities.

## 2.6 Conclusion

In this section, we presented some topics that compose the background of this research which include a brief discussion on agile development and agility assessment. We also presented a comprehensive analysis of the main approaches and tools focused on agility assessment. We analyzed 60 approaches and tools including text-based, graph-based, game-based, spreadsheet-based, and web-based approaches and confirmed the problems presented in Section 1.2 that are *Unclear assessment criteria selection*, *Unclear assessment criteria representation*, *Lack of support for adding new assessment criterion*, *Manual data collection and input*, *Lack of real-time assessment feedback*, and *Limited Scalability*. In the next chapter, we present the *AgileQube Approach*, an approach focused on agility assessment that address these limitations.

# Chapter 3

# *AgileQube Approach* Overview

*This chapter presents an overview of the AgileQube Approach proposed in this study that aims to automatically (or semi-automatically) detect agile smells in agile software projects.*

In Sections 1.2 and 1.3, we presented, respectively, the main problems of the existing agility assessment approaches and how the research proposal mitigates these problems. In a nutshell, we are adapting the *code smell* metaphor to the context of agility assessment and introducing the *agile smell* term to denote a situation that may be jeopardizing the adoption of an agile practice. The main goal of the research was summarized as: **propose an agility assessment approach that mitigates the problems P1 to P6 (described in Section 1.2), while providing a solid foundation for defining an infrastructure to support Agility Assessment**. This chapter presents an overview of the elements that compose the *AgileQube Approach* and how they are combined to fulfill the research goal.

The proposed approach is based on the method presented by MOHA *et al.* 2009 [139], DECOR. The DECOR method defines the phases for automatic detection of code smells. It is composed of four phases: *Description Analysis*, *Specification*, *Processing*, *Detection*, and *Validation*. We adapted this method to the context of agility assessment and designed an approach that we named *AgileQube Approach*. The annotated BPMN model presented in Figure 3.1 summarizes the proposed approach.

We organize the description of the *AgileQube Approach* overview in four sections: *Phases*, *Artifacts*, *Roles*, and *Computational Supporting Infrastructure*.

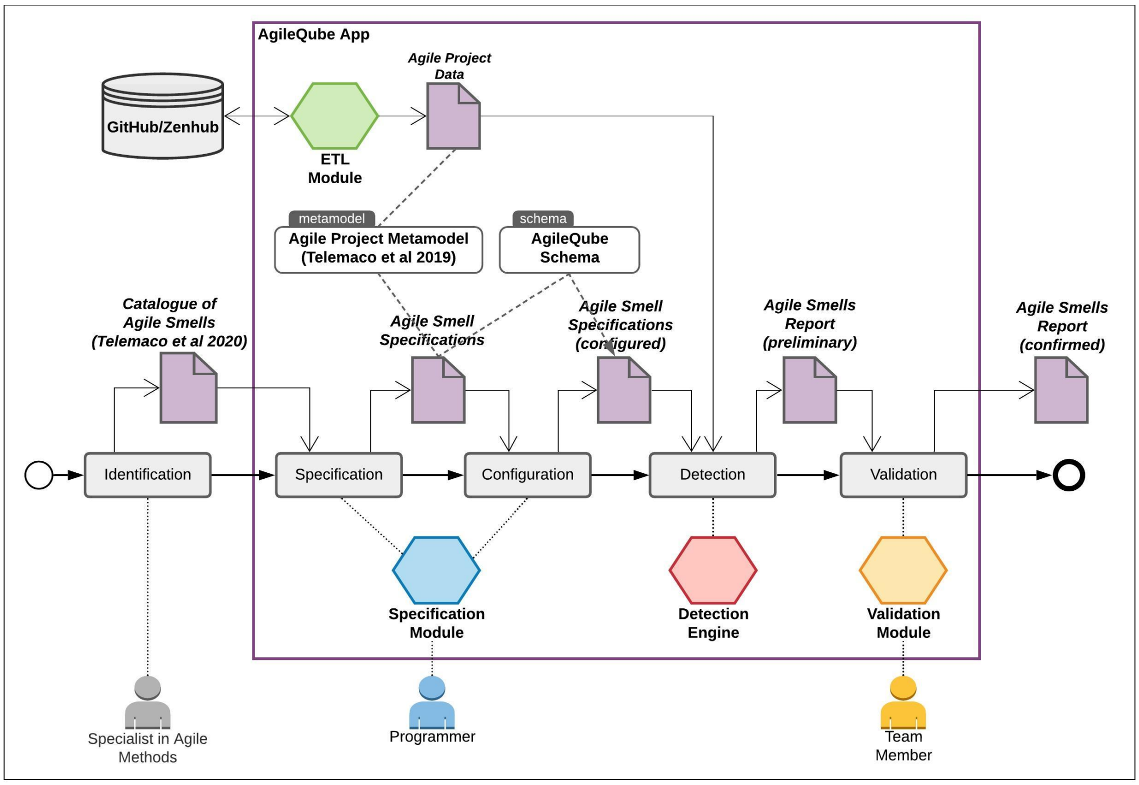

AgileQube App
GitHub/Zenhub
ETL Module
Agile Project Data
metamodel
Agile Project Metamodel (Telemaco et al 2019)
schema
AgileQube Schema
Catalogue of Agile Smells (Telemaco et al 2020)
Agile Smell Specifications
Agile Smell Specifications (configured)
Agile Smells Report (preliminary)
Agile Smells Report (confirmed)
Identification
Specification
Configuration
Detection
Validation
Specification Module
Detection Engine
Validation Module
Specialist in Agile Methods
Programmer
Team Member

Figure 3.1: *AgileQube Approach* overview

## 3.1 Phases

The *AgileQube Approach* has five phases: *Identification*, *Specification*, *Configuration*, *Detection*, and *Validation*.

1. **Identification**: the first phase of the method aims at identifying the agile smells that will be used in the next phases. Like DECOR, that advocates an explicit definition of the code smells, the *AgileQube Approach* also needs an explicit definition of the agile smells that will be specified, detected and validated in the following phases. This phase is specially important because the agile smells, in spite of being related to agile practices that are part of the routine of developers who work with agile methods, are usually not explicitly described. Thus, the interpretation of agile practices as compliance objectives and the subsequent identification as agile smells is necessary to the proposed approach. The *Identification* phase is performed by a *Specialist in Agile Development*. In this study, we performed this phase by identifying a set of agile smells through the following method: first, we conducted a literature review that investigated the state-of-the-art in software engineering and selected an initial set of agile smells. Second, the initial set of agile smells was the subject

of a survey with industry practitioners to confirm the agile smells. Third, we ranked the agile smells according to their relevance revealed in the survey and organized the agile smells in a structured catalogue. A detailed description of the literature review and the survey as well as the catalogue of the agile smells is presented in Chapter 4.

2. **Specification**: the second phase of the approach consists of translating the descriptions of the agile smells selected in the first phase into specifications. This phase is performed by a *Programmer* and is oriented by two elements: (a) a model to represent the data from an assessed agile software project; and (b) a mechanism to systematically specify an agile smell. Hence, to support the *Specification* phase, this study proposes a metamodel to represent agile software projects, that we named *Agile Project Metamodel*, and a schema to represent agile smells, that we named *Agile Smell Schema*. Chapters 5 and 6 present, respectively, the *Agile Project Metamodel* and the *Agile Smell Schema*.

3. **Configuration**: this phase aims at setting up the agile smell specifications according to the context of the project and organization. For example, the preferable duration of iterations (that may vary among organizations and even between the projects from the same company) is defined in this phase through parameters provided by the *Agile Smell Schema*. This phase is also performed by a *Programmer*.

4. **Detection**: the fourth phase of the approach is responsible for detecting the agile smells in an agile software project. This phase takes as input *(a)* data from a given agile software project (represented in the *Agile Project Metamodel*) and *(b)* a set of agile smell specifications (represented in the *Agile Smell Schema*). At the end of this phase, a set of candidate (or potential) agile smells is identified.

5. **Validation**: in the fifth and last phase, the agile smells identified in the previous phase can be individually assessed by a *Team Member* that may confirm or discard them.

The first step, *Identification*, was already executed as part of this research (see Chapter 4) and as a result, organizations willing to apply the *AgileQube Approach* can skip this step and use the *Catalogue of Agile Smells* as input for the next steps. It is important to note that this step may be executed in different occasions as, for example, during a literature review to update the catalogue of agile smells. The steps *Specification* and *Configuration* should be executed only during the approach setup and when specifying a new agile smell. The last two steps, *Detection* and *Validation*, are always executed during an agility assessment.

## 3.2 Artifacts

The approach prescribes 6 types of artifacts:

1. ***Catalogue of Agile Smells***: the set of agile smells that is produced in the *Identification* phase and used as input for the *Specification* phase.
2. ***Agile Smell Specifications***: the agile smell specifications represented in the *Agile Smell Schema* that is produced in the *Specification* phase and used as input for the *Configuration* phase.
3. ***Agile Smell Specifications (configured)***: the agile smell specifications configured to the context of the project and organization. This artifact is the output of the *Configuration* phase and is used as input for the *Detection* phase.
4. ***Agile Project Data***: data from the assessed agile software project represented in the *Agile Project Metamodel*. This artifact is collected by the *ETL Module* and used as input for the *Detection* phase.
5. ***Agile Smells Report (preliminary)***: a preliminary version of the *Agile Smells Report* indicating the agile smell detected in the assessed agile software project. This report is the output of the *Detection* phase and is used as input for the *Validation* phase.
6. ***Agile Smells Report***: a confirmed version of the report that is produced in the *Validation* phase.

## 3.3 Roles

The *AgileQube Approach* has three roles: *Specialist in Agile Methods*, *Programmer*, and *Team Member*.

1. **Specialist in Agile Methods**: this role acts in the *Identification* phase by selecting the agile smells that will be used in following phases of the approach.

Ideally, this role should be performed by someone who masters agile development.

2. **Programmer**: this role is responsible for coding the agile smells using the *Agile Smell Schema*, i.e, translating the description of agile smell into specifications in the *Specification* phase. The *Programmer* also configures the agile smells specifications to the context of the project and organization in the *Configuration* phase.

3. **Team Member**: this role acts in the *Validation* phase by confirming or rejecting the agile smells identified by the *Detection Module.* The validation of the agile smells should be performed by someone involved in the project (project manager, scrum master, developer, etc).

## 3.4 Computational Supporting Infrastructure

The computational supporting infrastructure, that we named *AgileQube App*, was designed to aid the execution of the *Specification*, *Configuration*, *Detection* and *Validation* phases and is composed of the *Specification Module*, *ETL Module*, *Detection Engine*, and *Validation Module* components.

1. ***Specification Module***: this module aids the *Programmer* performing the *Specification* and *Configuration* phases. It provides the interfaces that allow specifying and configuring an agile smell using the *Agile Smell Schema*.

2. ***ETL Module***: the *ETL Module* (that stands for extracting, transforming and loading) supports the *Detection* phase by loading the data related to the agile project from a project management platform and transforming the loaded data into the model supported by the *Detection Engine* (*Agile Project Metamodel*). The *ETL Module* currently is able to load data from the *ZenHub* platform and transform data to the *Agile Project Metamodel* format.

3. ***Detection Engine***: this component plays an important role in the approach and is responsible for detecting the agile smells in an agile project. To achieve this goal, it receives two inputs: *(a)* data from an agile project (represented in the *Agile Project Metamodel*) and *(b)* a set of agile smell specifications (represented in the *Agile Smell Schema*).

4. ***Validation Module***: this module aids a *Team Member* performing the *Validation* phases. It provides the interfaces to show details of the identified agile smells and allow the *Team Member* to confirm or reject the agile smells.

The *AgileQube App* and its components are presented in Chapter 7.

## 3.5 Conclusion

This chapter was intended to provide an overview of the *AgileQube Approach*, an approach to automatically (or semi-automatically) detect agile smells in agile software projects. This overview was organized in four sections: *Phases*, *Artifacts*, *Roles*, and *Supporting Infrastructure.*

The remainder of this thesis describes in more details the main components that compose the approach. Chapter 4 describes how the *Identification* phase was conducted and presents the *Catalogue of Agile Smells*. Chapter 5 presents the *Agile Project Metamodel* that is used in the *Specification*, *Configuration* and *Detection* phases. The schema to represent the agile smells, the *Agile Smell Schema*, is presented in Chapter 6. The *AgileQube App* is presented in Chapter 7.

# Chapter 4

# *Catalogue of Agile Smells*

*This chapter presents the Catalogue of Agile Smells and is organized in the following sections: Introduction, Research Methodology, Literature Review, Survey, Consolidation Phase, Threats to validity, and Conclusion.*

## 4.1 Introduction

The *Catalogue of Agile Smells* is one of the contributions of this study and is directly answering the goal G1 (*Define a catalogue of agile smells*).

The catalogue was elaborated in the *Identification* phase of the *AgileQube Approach* and organizes a set of agile smells extracted from the specialized literature and confirmed by the industry. The catalogue gives a clear definition of the agile smells indicating, for each agile smell, a name, a description, which agile practices motivated the agile smell, and at least one identification strategy that guides the specification of the agile smell. The *Catalogue of Agile Smells* is an important contribution because, as we have observed (initially through a preliminary *Ad hoc* literature review and later confirmed through a literature review) there are few studies that explicitly describe agile smells or agility assessment criteria. 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* (BECK *et al.* 2001 [53]), for example, proposed a set of values and principles that have inspired many agile methods. However, the manifesto does not indicate how to verify that such values and principles are being applied properly. 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 (ALLIANCE 2016 [21]), XP (CUNNINGHAM 1999 [57]), Crystal Family Methods (COCKBURN 2002 [23]), and Open Up (FOUNDATION 2012 [140]) 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 development, or (c) to inspire discussions among the team in retrospective meetings.

To determine the agile smells, this study tries to answer two 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 applying 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 the agile smell specification less subjective and less compromised by the *Programmer* bias in the *Specification* phase. Hence, the identification strategies guide the coding of the agile smell specifications that will be ultimately used by the *Detection Engine* to identify the agile smells. We sought to answer RQ2 by proposing at least one identification strategy for each agile smell. By answering these two questions, we provide a baseline to support agility assessment at organizational and project levels.

## 4.2 Research Methodology

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 (SPÍNOLA *et al.* 2008 [141]) and consists of four steps divided into three phases as depicted in Figure 4.1:

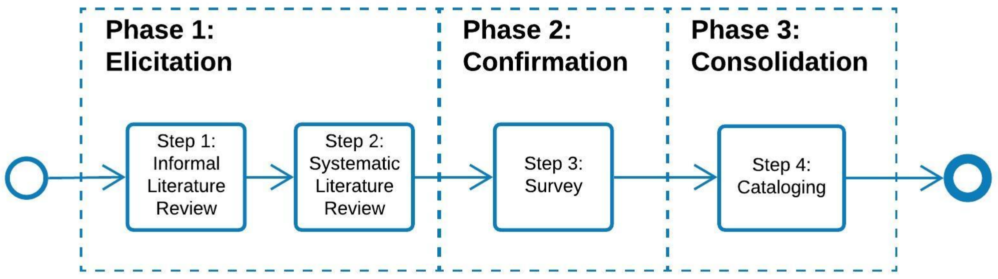


Figure 4.1: The research methodology organized in three phases and four steps.

*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 literature review protocol; and (2) A literature review that was planned and executed to identify a set of agile smells. The literature review design details, the mechanisms and collected data and the set of identified agile smells are described in Section 4.3.

*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 4.4

*Phase 3 - Consolidation*: In this phase, the most relevant agile smells were organized in a structured format named the *Catalogue of Agile Smells*. This catalogue is presented in Section 4.5.

## 4.3 Literature Review

In the elicitation phase, a literature review (LR) 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 LR was based on the method proposed by KITCHENHAM and CHARTERS 2007 [142] and consists of three main phases: *planning*, *execution* and *reporting*.

### 4.3.1 Literature Review Planning

#### 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 LR 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 LR were derived from the RQ1 and RQ2 (presented in Section 4.1) and can be summarized as:

**LR-RQ1:** *What are the practices that impair the proper adoption of agile development?*

**LR-RQ2:** *How can we identify the occurrence of these practices?*

The scope of this review was defined based on the population, intervention, comparison and outcome (PICO (PAI *et al.* 2004 [143])) approach. The *Population* denotes 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 *Ad hoc* literature review were used as control:

1. 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, Washington, DC, USA, 2001. IEEE Computer Society [144];

2. LINDVALL, M., BASILI, V. R., BOEHM, B. W., et al. 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*, London, UK, UK, 2002. Springer-Verlag [145];

3. 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 [66]

.

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.* 2004 [143]), is presented in Listing 4.1:

```
(
    '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 'technic' OR 'techniques'  OR
    'classification'
)
```

Listing 4.1: Agile smells literature review search string base

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.

**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. Since the values and principles proposed by the *Manifesto for Agile Development* and the methodologies derived from the manifesto are described in a vague way (TSOURVELOUDIS and VALAVANIS 2002 [146], GILL and HENDERSON-SELLERS 2006 [147]), 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.

### 4.3.2 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 (LI *et al.* 2010 [148]), 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

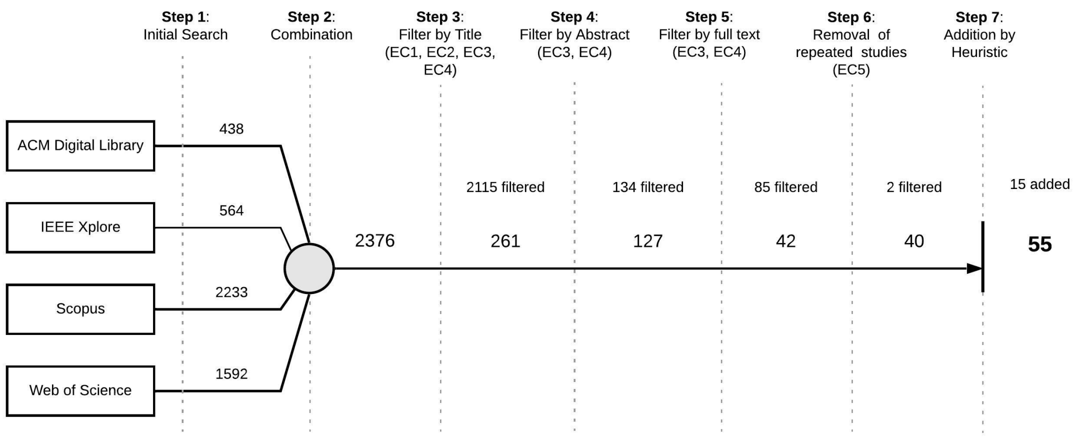
EC3 and EC4 by reading the abstract of the studies. At the end of this step, 127 studies remained.

- *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, including grey literature sources, totaling 55 studies. These studies were added manually, based on our background knowledge. Appendix C shows the final list of studies considered in this literature review.

Figure 4.2 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.



**Step 1**: Initial Search
**Step 2**: Combination
**Step 3**: Filter by Title (EC1, EC2, EC3, EC4)
**Step 4**: Filter by Abstract (EC3, EC4)
**Step 5**: Filter by full text (EC3, EC4)
**Step 6**: Removal of repeated studies (EC5)
**Step 7**: Addition by Heuristic
ACM Digital Library
438
IEEE Xplore
564
Scopus
2233
Web of Science
1592
2376
2115 filtered
261
134 filtered
127
85 filtered
42
2 filtered
40
15 added
55

Figure 4.2: The process of primary study selection

### 4.3.3 Literature Review Reporting

During the LR, 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 LR 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 LR-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: [58], [63], [57], [55], [56], [62], [24], [61], [149], [64], [150], [59], [23], [151], [152], [66], [153], [154], [155], [156], [60], [21], [157], [158].

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: [63], [57], [55], [56], [149], [150], [23], [152], [66], [159], [153], [4], [160], [154], [161], [162], [163], [5], [157], [164], [165].

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 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: [58], [24], [144], [59], [23], [151], [159], [153], [4], [154], [155], [60], [21], [158].

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: [149], [151], [159], [153], [60]. [162], [157].

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: [58], [59], [151], [166], [152], [66], [4], [154], [21], [157], [165].

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: [72], [4], [167], [60].

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: [57], [55], [56], [149], [150], [59], [168], [169], [170], [171], [152], [159], [153], [4], [160], [154], [155], [60], [17], [21].

8. ***Goals Not Defined or Poorly 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 or Poorly 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: [58], [63], [62], [24], [61], [144], [64], [59], [23], [151], [153], [156], [17], [21]. [157].

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: [170], [72], [167], [60], [17], [165].

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: [58], [63], [57], [55], [56], [62], [24], [61], [149], [64], [150], [59], [23], [151], [66], [153], [154], [156], [60], [21], [158].

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: [58], [63], [57], [55], [56], [62], [24], [61], [149], [64], [150], [59], [23], [151], [172], [169], [170], [152], [66], [173], [159], [153], [4], [167], [154], [155], [60], [21], [161], [162], [163], [5], [157], [164], [165].

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: [58], [63], [62], [24], [61], [64], [59], [23], [151], [152], [159], [153] [4], [154], [155], [156], [60], [161], [5], [162], [163], [157], [158], [164], [165].

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: [58], [63], [62], [24], [61], [64], [59], [23], [151], [152], [173], [159], [153], [4], [154], [60], [21], [161], [163], [5], [157], [158], [164].

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: [152], [72], [4], [167], [60], [157].

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: [58], [62], [61], [145], [59], [168], [170], [153], [154], [155], [60], [21], [157], [158].

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: [151], [72], [4], [167], [60],

[162], [164].

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: [58], [63], [57], [55], [56], [62], [24], [61], [149], [64], [150], [59], [23], [151], [153], [4], [154], [156], [17], [21], [157], [158], [164].

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: [151], [168], [4], [160], [60], [157], [158], [165].

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: [151], [72], [4], [167], [60]. [17], [157].

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: [149], [151], [153], [4], [156], [60], [17], [157],

[164].

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

1. **Identification Strategy for the *Complex Tasks* agile smell**: A strategy to identify the presence of the agile smell *Complex Tasks* 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 4.5.

## 4.4 Survey

In order to confirm the results from the literature review, we conducted a survey with practitioners based on semi-structured interviews (KAJORNBOON 2005 [174]). The remainder of this section presents the survey that was based on the protocol proposed by OISHI 2003 [175]. The survey was divided into three phases: *planning*, *execution* and *reporting.*

### 4.4.1 Survey Planning

#### Aim and Research Questions

The aim of the survey was to evaluate the relevance of the identified agile smells for purposes of 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?*

### 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 4.3.3) in the following structure: *Name*, *Short Description* and *Identification Strategy*. The participants answered two questions for each agile smell:

**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?*

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 SQ1 and SQ2 as shown in Figure 4.3. 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.

$$\boxed{relevanceByParticipant(p) = answerQuestion1(p) + answerQuestion2(p)}$$

Figure 4.3: Formula of agile smell relevance by participant.

The final relevance of an agile smell is the sum of the relevance for all participants as illustrated in the formula presented in Figure 4.4.

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

Figure 4.4: Formula of final agile smell relevance.

### Participants Selection

We applied a convenience sampling approach (GHAZI *et al.* 2017 [176]) 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.

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.

Table 4.1: 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% |

### 4.4.2 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 4.1 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 3 participants have more than 21 years of professional experience. Regarding the geographic distribution, 16 participants are from Brazil and 4 from Canada.

### 4.4.3 Survey Reporting

In the last phase, the data collected in the survey were organized, tabulated and analyzed. Table 4.2 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 *R* (that stands for *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 presented in Figure 4.4.

Table 4.2: Summary of data collected and analyzed in the survey.

| **R** | **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 | 3 | 2 | 3 | 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 or Poorly 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 | 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** |

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*.

### 4.4.4 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 4.2, they were all considered at least *Slightly relevant* to all the participants.

Figure 4.5 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.

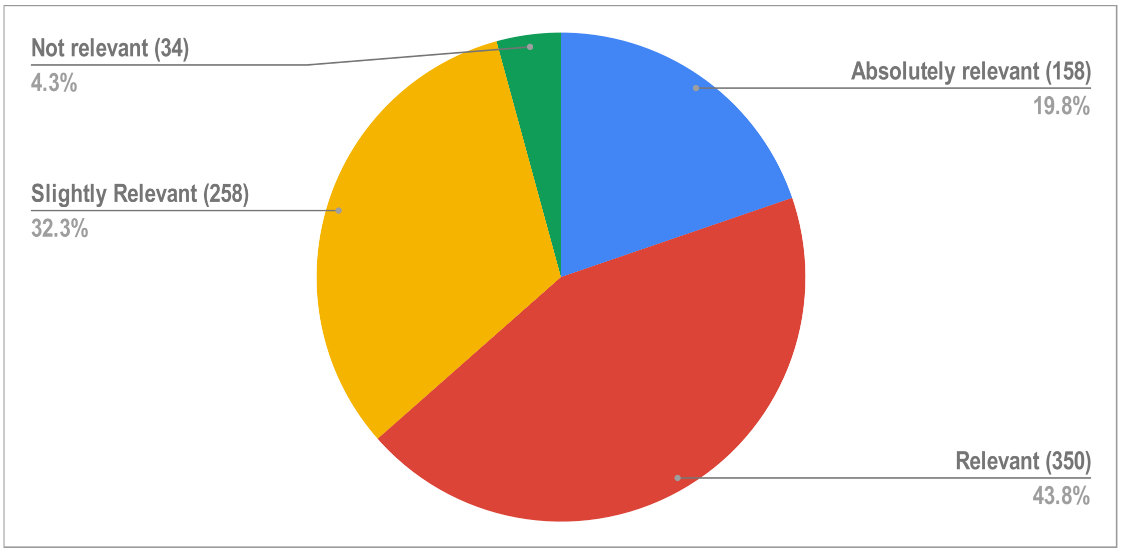


Figure 4.5: Distribution of the degree of relevance according to the survey.

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

Table 4.3: Agile smell template.

| **Name**: **Agile smell 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 |

bottom of the ranking vary from 57 to 49. This difference between the degree of relevance illustrates that the agile smells may impact the adoption of agile methods in different levels.

## 4.5 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 (GAMMA *et al.* 1995 [177]) and shown in Table 4.3.

The *Name* section indicates 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)

(COHEN *et al.* 1999 [178]). The *Identification Strategy* and *Parameters* sections describe, respectively, strategies to detect the occurrence of the agile smell in real agile projects and their corresponding parameters. These sections are designed to support the *Specification* phase of the *AgileQube Approach*.

We have selected the 10 highest ranked agile smells to present in Appendix D (Tables D.1 to D.10).

## 4.6 Threats to validity

This section discusses the threats to the validity of this study 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 practitioners that usually do not publish in academic forums (GLASS and DEMARCO 2006 [179]), 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 "agile 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. Another threat in this category came from the fact we are using the data collected from Likert scale as continuous data as presented in Figures 4.3 and 4.4. Appendix B presents a discussion on this issue.

*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.

## 4.7 Conclusion

This chapter presented the *Catalogue of Agile Smells*, which is one of the contributions of this research and that supports the goal G1 (*Define a catalogue of agile smells*). The presented catalogue aims at: *(a)* presenting a set of agile smells (i.e. practices that may impair the proper adoption of agile development); *(b)* relating these agile smells with the agile practices and methods that motivated them; and *(c)* proposing strategies to identify the occurrence of such agile smells in an agile software project.

The catalogue was produced in the *Identification* phase of the *AgileQube Approach* and its elaboration followed a three-phase methodology: *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, the set of selected agile smells was the subject of a survey that aimed at characterizing the smells according to their relevance. Finally, in the Consolidation phase, the data collected in the survey were analyzed and the most relevant agile smells were organized as a catalogue. One threat to this analysis came from the fact we treated the data collected in the survey (a 4-level Likert scale) as a continuous scale. Appendix B discusses this threat in more detail and presents another analysis of the data collected in the survey.

# Chapter 5

# *Agile Project Metamodel*

*This chapter presents the Agile Project Metamodel, a metamodel used to represent agile software projects and support the Specification and Detection phases of the AgileQube Approach. The chapter is organized in the following structure: Introduction, Concepts necessary to represent agile projects, Literature Review, The Agile Project Metamodel, and Conclusion.*

## 5.1 Introduction

The *Agile Project Metamodel* is one of the components that compose the *AgileQube Approach* and is responsible for representing the data from the assessed agile project. This contribution aids this research in reaching the goals G2 (*Define an agile smell representation approach*) and G3 (*Define a data extraction approach*).

As depicted in Figure 5.1, the metamodel plays a key role in the approach and is used in the *Specification* and *Detection* phases. In the *Specification* phase, the elements of the *Agile Project Metamodel* are directly referenced in the agile smell specifications (the agile smell specification will be discussed in Chapter 6). In the *Detection* phase, the *ETL Module* extracts data from a *Project Management Tool*, transforms the extracted data to the *Agile Project Metamodel* format and sends the resulting model, along with the agile smell specifications, to the *Detection Engine*.

A preliminary investigation conducted in this study revealed that existing software development metamodels fail to represent all the elements the proposed approach needs. The existing metamodels usually focus on representing elements related to software process definition and neglect aspects related to agile project execution such as *Iterations*, *Deliveries*, and *Task Instances*.

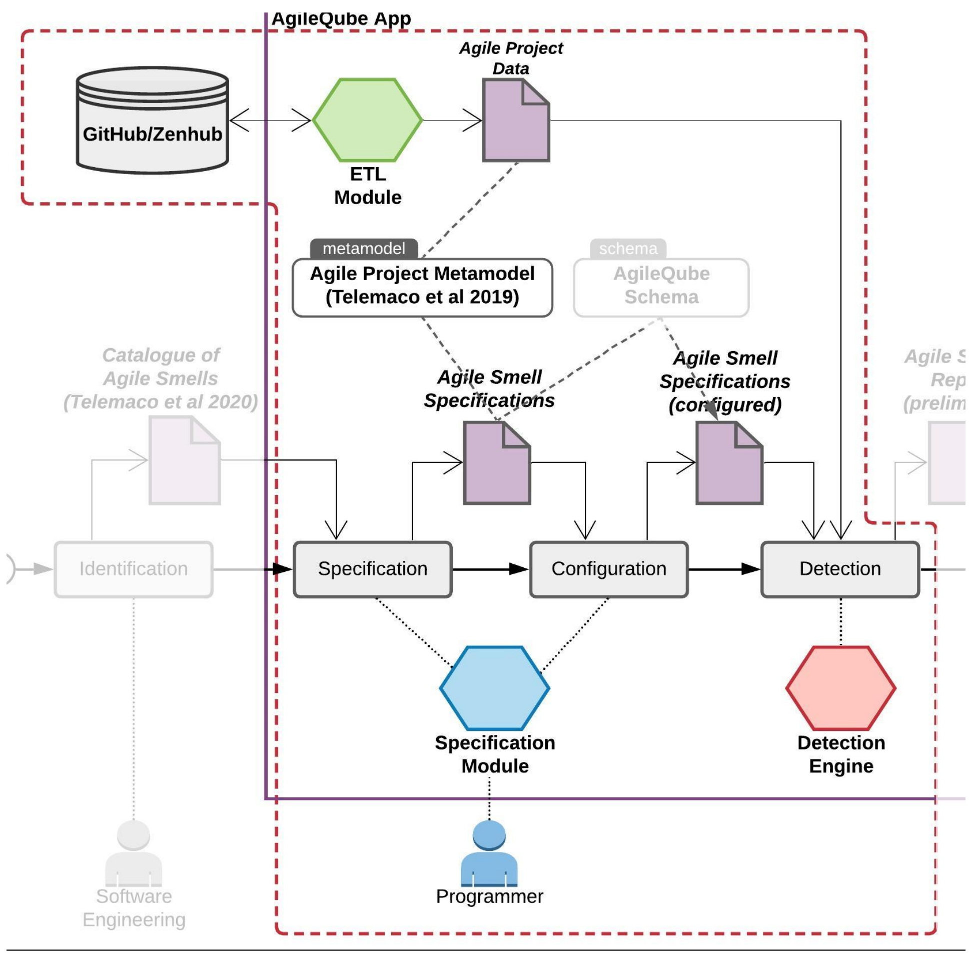


Figure 5.1: *Agile Project Metamodel* overview

This chapter aims at presenting the *Agile Project Metamodel* that composes the *AgileQube Approach*. We conducted this study in three steps: First, we identified the set of concepts, entities, and information needed in the approach (i.e., for purposes of agility assessment). Second, we investigated how existing software development metamodels represent these elements. Third, we proposed the *Agile Project Metamodel* which contains all the elements identified in the first step.

Two research questions guided this study:

**RQ1:** *What are the main concepts and information necessary to represent an agile software project?*

**RQ2:** *Which software development metamodels contain or partially contain the information necessary to represent an agile software development project?*

To answer RQ1, we used the *Catalogue of Agile Smells* presented in Chapter 4 as source for defining the required elements. A literature review was conducted to answer RQ2 and revealed that there is no metamodel that fully represents the information identified in RQ1. To fill this gap, we investigated how the metamodels can be combined and extended.

As a result, this study proposes the *Agile Project Metamodel*, a metamodel that combines elements from existing metamodels with elements neglected by these metamodels. The metamodel is organized in two perspectives, *Process Definition* and *Process Execution* (as proposed by SANTOS 2019 [180]), and includes entities such as *Project*, *Iteration*, and *Task* as core concepts.

## 5.2 Concepts necessary to represent agile projects

The goal of this phase was to answer the research question RQ1 and identify a set of concepts necessary to represent agile software projects. We used as source of information the *Catalogue of Agile Smells* presented in Chapter 4.

To identify a set of relevant concepts, we applied an iterative and incremental method and for each agile smell, we analyzed its description and noted the main concepts as *codes*. As more data were collected, and re-reviewed, codes were grouped into concepts. Repeated tagged-with-codes concepts, concepts or elements became apparent. At the end of this method, the following concepts composed the basis of the proposed metamodel: *Project*, *Iteration*, and *Task*.

To identify a set of relevant concepts, we applied the following iterative and incremental method: for each agile smell, we analyzed its description and noted the main concepts as *potential concepts*. As more data were collected, and re-reviewed, the potential concepts were grouped into relevant concepts. At the end of this method, the following concepts composed the basis of the proposed metamodel: *Project*, *Iteration*, and *Task*.

**Project:** The *Project* concept is a core entity in the metamodel and all other entities are in some way related to it. Some agile smells directly mention this concept. For example, the *Goals Not Defined or Poorly Defined* agile smell states that:

*The **Project's** and Iterations' goals should be clear and well-defined.*

Other agile smells mention the *Project* concept implicitly, as for example, the *Absence of Timeboxed Iteration* agile smell that defines:

> *All Iterations [of the* ***Project****] should have a fixed time duration.*

**Iteration:** The *Iteration* element represents a development cycle that usually produces a portion of the software. Several agile smells refer to the *Iteration* concept. For example, the *Absence of Timeboxed Iteration* agile smell states:

> *All* ***Iterations*** *should have a fixed time duration. Thus, an* ***Iteration*** *should not be extended or shortened to fit planned or unplanned features.*

Another example is the *Lower Priority Tasks Executed First* agile smell that defines:

> *The team should deliver a new version of the software at the end of each* ***Iteration***

**Task:** The *Task* element denotes a unit of work. It typically has information about its execution such as start and end dates, complexity, duration, who performed the task and which artifacts were consumed/produced. The *Task* concept was extracted from agile smells such as *Lower Priority Tasks Executed First* that states:

> *The Team must work on the highest priority* ***Tasks*** *first.*

Another example is the *Complex Tasks* agile smell that states

*Complex* ***Tasks*** *should be avoided. During Iteration planning, the Team should try to break complex* ***Tasks*** *into simpler* ***Tasks****.*

It is important to note that a *Task* element is an instance of a *Task Definition* defined by a software development process. To avoid confusion, in the remainder of this chapter we will use the term *Task* to denote a task instance and *TaskDef* to denote a task definition.

## 5.3 Literature Review

The literature review (LR) conducted in this study aimed at exploring the existing body of knowledge and investigating how software development metamodels proposed in indexed studies represent (or partially represent) the concepts identified in the previous phase. As this is an exploratory study designed to organize a research area (metamodels to support the representation of agile software projects), there is no baseline for comparison of the results obtained (TRAVASSOS *et al.* 2008 [181]). The methodology of the LR was based on the method proposed by KITCHENHAM and CHARTERS 2007 [142] and consists of three main phases: *planning*, *execution* and *reporting*.

### 5.3.1 Literature Review Planning

The research question for this LR was derived from the main RQ2 and can be summarized as: *Which Software Development Metamodels contain (or partially contain) the information necessary to represent an agile software development project?*

The scope of this literature review was defined according to the population, intervention, comparison and outcome PICO (PAI *et al.* 2004 [143]) approach. The *Population* is the set of software development projects. The *Intervention* is the agile methodologies and software processes. There is no *Comparison*. The *Outcome* is a set of metamodels to represent software development. Three papers obtained from a previous conventional literature review were used as control:

1. STEENWEG, R., KUHRMANN, M., MÉNDEZ FERNÁNDEZ, D. Software engineering process metamodels–a literature review, *Technische Universität München, Tech. Rep. TUM-I1220*, 2012 [182]

2. BENDRAOU, R., JEZEQUEL, J.-M., GERVAIS, M.-P., et al. A comparison of six UML-based languages for software process modeling, *IEEE Transactions on Software Engineering*, v. 36, n. 5, pp. 662–675, sep 2010 [183]

3. HENDERSON-SELLERS, B., GONZALEZ-PEREZ, C. A comparison of four process metamodels and the creation of a new generic standard, *Information and Software Technology*, v. 47, n. 1, pp. 49 – 65, 2005. ISSN: 0950-5849. doi: https://doi.org/10.1016/j.infsof.2004.06.001 [184]

The keywords for *Population* are "software projects", "software systems", "software development" and "software engineering". The keywords for *Intervention* are "agile methods", "agile processes" and "software processes". The keyword for *Outcome* is "metamodel". The search string used as the basis for all search engines, structured according to PAI *et al.* 2004 [143], is presented in the Listing 5.1:

```
(
   'software project'  OR  'software systems'  OR
   'software development'  OR  'software engineering'
)
AND
(
    'agile methods'  OR  'agile processes'  OR
    'agile approaches'  OR  'software processes'
)
AND
(
    'metamodel'
)
```

Listing 5.1: Software development literature review search string base

We chose an incremental approach for the study selection. The final set of publications is a combination of automated search strategies and "snow-balling" procedures. The sources were collected from the following digital databases, including the conferences, journals and technical reports indexed by IeeeXplore, Web of Science, Scopus and ACM digital library.

The set of formal literature studies includes all articles returned by the protocol that meets the following inclusion criterion (IC): Documents must address one or more software development/process metamodel; Publications that satisfy at least one of the following exclusion criteria (EC) were excluded: (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, that is the one explaining the study in greatest detail, will be used and the others will be excluded.

The following information was extracted from each paper selected after running the selection process: *document title*, *author(s)*, *source*, *year of publication*, *concepts* and *elements* described in the metamodels.

The overall goal of the literature review was to select contributions that propose any metamodel related to software process and analyze the metamodels in order to investigate how they represent the concepts presented in Section 5.2. Other characteristics, such as the metamodel relevance, adoption by industry, process ecosystem and supportive tools were not considered in this study. HENDERSON-SELLERS and GONZALEZ-PEREZ 2005 [184], KUHRMANN *et al.* 2013 [185] and BENDRAOU *et al.* 2010 [183] are some examples of comprehensive studies about Software Process Metamodel Languages (SPML).

### 5.3.2 Literature Review Execution

The search with the aforementioned engines returned 119 references, published between 2001 and 2017. All the controls were retrieved. Repetitions were eliminated and after the application of the inclusion and exclusion criteria, the number of papers selected for full reading was 14. Appendix E shows the final list of studies considered in this literature review. The selected documents were used to extract the main concepts of the metamodel.

### 5.3.3 Literature Review Reporting

In the reporting phase, we analyzed the selected studies and investigated how the metamodels proposed represent the concepts presented in Section 5.2. Following the criteria established in the research protocol, 12 metamodels were selected from 15 documents. These metamodels in alphabetical order are:

1. *APM$^3$* (SADI and RAMSIN 2009 [186])
2. *Ayed's approach* (AYED *et al.* 2012 [187])
3. *Chou's approach* (CHOU 2002 [188])
4. *DiNitto's approach* (NITTO *et al.* 2002 [189])
5. *ISO/IEC 24744 (SEMDM)* (ISO 24744:2014 [190])
6. *MetaMe* (ENGELS and SAUER 2010 [191])
7. *OOSPICE* (GONZALEZ-PEREZ *et al.* 2005 [192])

8. *OPF* (FIRESMITH and HENDERSON-SELLERS 2002 [193])
9. *PROMENADE* (FRANCH and M. RIB 1999 [194])
10. *SPEM* (OMG 2008 [195])
11. *UML4SPM* (BENDRAOU *et al.* 2005 [196])
12. *V-Modell XT* (TERNITÉ and KUHRMANN 2009 [197])

One approach initially selected was discarded, the *LiveNet* approach (referenced by HENDERSON-SELLERS and GONZALEZ-PEREZ [184]) was not available.

One of the most well-established metamodels for specifying Software Process is SPEM (*Software & Systems Process Engineering Metamodel*) (OMG 2008 [195]). The last specification, SPEM 2.0, was launched in 2008 by the OMG Group and is focused on defining software and systems development processes and their components without adding specific features for particular development domains or disciplines such as project management. SPEM aims at enabling the specification of a large range of development methods and processes of different styles, levels of formalism, life-cycle models and areas without imposing predefined modeling concepts. To achieve this, the SPEM specification is divided into two main packages: *Method Content* and *Process with Methods*. The *Method Content* package defines the core concepts such as *Task*, *Role* and *Work Product*, while the *Process with Methods* package supports the definition of software process models as nested activities that embody essentially the predefined elements in the *Method Content* package.

The main limitation of SPEM in representing agile software projects is that the metamodel is mainly focused on software process definition and neglects project execution elements, HENDERSON-SELLERS and GONZALEZ-PEREZ 2005 [184] call *methodology layer* and *project layer*. For example, the *Task* element in SPEM represents a task definition and not a task instance. This element has only fields to describe general characteristics of the task such as description, predecessor tasks and successor tasks. There is no execution information in its element.

Other analyzed metamodels such as *Chou's approach*, *DiNitto's approach*, *MetaMe*, *OOSPICE*, *OPF*, *PROMENADE*, *UML4SPM* and *V-Modell XT* are similar to SPEM in this aspect. They focus on the methodology layer and neglect the representation of the project layer. These approaches have the elements *Activity*, *Role*, *Artifact* and *Tool* as a basis for software process definition. No or little information about process execution is addressed by these metamodels. The exception is the *MetaME* metamodel that represents the concept *Project*. The metamodels *APM$^3$* and *Ayed's approach*, on the other hand, focus on the project layer of Software Development and do not represent elements related to Process Definition. The only metamodel that supports the representation of both process and project layer is the *ISO/IEC 24744 (SEMDM)* metamodel that has elements corresponding to the concepts *Iteration*, *Task*, *Worker*, *TaskDef*, *RoleDef* and *ArtifactDef*.

Tables 5.1 and 5.2 present a summary of the analyzed software development metamodels.

Table 5.1: Software Process Metamodels and their corresponding elements (part1)

| | **APM$^3$** | **Ayed** | **Chou** | **Di Nittos** | **ISO 24744** | **MetaME** |
|---|---|---|---|---|---|---|
| **Project** | - | - | - | - | - | Project |
| **Iteration** | Stage | Stage | - | - | StageWithDuration | - |
| **Iteration Interval** | - | - | - | - | - | - |
| **Case** | - | - | - | - | - | - |
| **Task** | Work-unit | Task | - | - | WorkUnit, Task | - |
| **Delivery** | Product | Deliverable | - | - | - | - |
| **Delivery Interval** | - | - | - | - | - | - |
| **Worker** | Role, Responsibility | Producer | - | - | Producer, Person, Team, Role | - |
| **TaskDef** | - | - | Activity | Activity, Humam Activity | WorkUnitKind | Process, Activity, Task, ActionStep |
| **RoleDef** | - | - | - | - | ProducerKind, RoleKind | Role |
| **ArtifactDef** | - | - | Document | Artifact | WorkProductKind | Artifact |

Table 5.2: Software Process Metamodels and their corresponding elements (part2)

| | **OOSPICE** | **OPF** | **Promenade** | **SPEM** | **UML4SPM** | **V-Modell XT** |
|---|---|---|---|---|---|---|
| **Project** | - | - | - | - | - | - |
| **Iteration** | - | - | - | - | - | - |
| **Iteration Interval** | - | - | - | - | - | - |
| **Case** | - | - | - | - | - | - |
| **Task** | - | - | - | - | - | - |
| **Delivery** | - | - | - | - | - | - |
| **Delivery Interval** | - | - | - | - | - | - |
| **Worker** | - | - | - | - | - | - |
| **TaskDef** | Technique, Task, Action | WorkUnit, Activity, Task, Subtask, Technique | Task | Activity, Task, Definition | Software Activity | Activity, SubActivity, Step |
| **RoleDef** | - | Role | Role | RoleUse, RoleDefinition | Responsible Role | Role |
| **ArtifactDef** | WorkProduct | WorkProduct | Document | WorkProduct | WorkProduct | Product, Subject, Topic |

We analyzed 12 software development metamodels. Most approaches focus on enabling the definition of Software Processes and Methods and neglected the representation of elements related to the execution of a Software Process.

## 5.4 The *Agile Project Metamodel*

After identifying a set of elements necessary to represent an agile project for the purposes of agility assessment (Section 5.2) and conducting a literature review to investigate how existing software development metamodels can be used to represent such elements (Section 5.3), we observed a need for a metamodel that fully supports the representation of an agile software project.

To fill this gap, we proposed a software development metamodel, that we named *Agile Project Metamodel* (APMM), to represent all the elements selected in the first step of this study. Figure 5.2 shows the resulting metamodel that is organized in the two perspectives proposed by SANTOS 2019 [180]: *Process Definition* and *Project Execution*. The *Process Definition* layer focuses on representing static concepts of the software process and the *Process Execution* layers contain elements that represent dynamic concepts of the agile project.

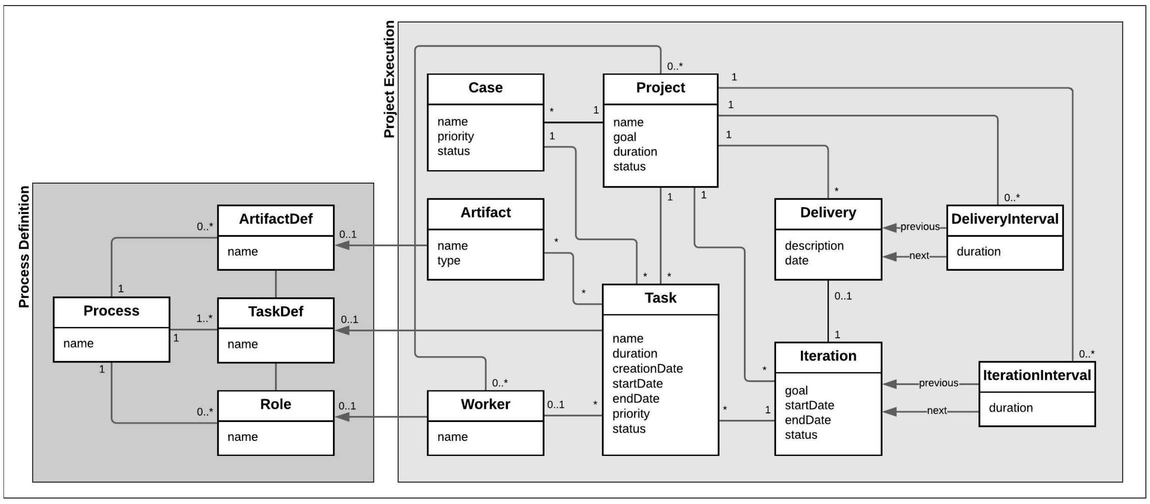


Figure 5.2: *Agile Project Metamodel* (APMM)

### Process Definition Perspective

- **Process**: The software processes that guide the *Project*. A *Process* has three collections: *ArtifactDefs*, *TaskDefs*, and *Roles*. Note that a *Project* can have tasks from many different *Processes*.

- **TaskDef**: The specification of a *Task* from the perspective of process definition. Some metamodels such as SPEM (OMG 2008 [198], FIRESMITH and HENDERSON-SELLERS 2002 [193]) use the term *Activity* or *WorkUnit*.

- **ArtifactDef**: The specification of an *Artifact* from the perspective of process definition.

- **Role**: The specification of a *Role* from the perspective of process definition. A *Role* defines the responsibilities of workers that take part in the software process.

**Project Execution Perspective**

- **Project**: The core entity of the model represents the software project. The *Project* is associated with *Iterations*, *Cases*, *Tasks*, *Deliveries* and *Delivery Intervals*.

- **Iteration**: An *Iteration* denotes a cycle of development that typically produces a part of the software. It has a duration, a start and an end date, a status and other fields. Some agile methods such as Scrum (ALLIANCE 2016 [21]) use the term *Sprint*. FDD methods (LUCA 1999 [62]) use the term *Cycle*. As the main associations, an *Iteration* belongs to a *Project* and has a collection of *Tasks*.

- **IterationInterval**: The interval between two consecutive *Iterations*.

- **Case**: A *Case* represents a coherent subset of tasks. For example, in a project where the requirement is oriented by Use Cases, a *Case* denotes the set of tasks that compose a given Use Case. A *Case* can also be a software feature, a software module, a change request, a bug, etc. A *Case* usually has its own flow, which typically has requirement, development and test tasks. A *Case* may be divided among many *Iterations*.

- **Task**: A *Task* represents a unit of work that is performed during software development. It may be associated with a *Case*, associated with an *Iteration* and assigned to a *Worker*. A *Task* in this perspective has information related to process execution, such as start and end dates, estimated effort, status, time spent on task and worker assigned. A *Task* could also be associated with a *TaskDef*, and, in this case, the *Task* is an instance of the *TaskDef*.

- **Delivery**: A *Delivery* represents the software product delivered at the end of an *Iteration.*
- **DeliveryInterval**: The interval between two consecutive *Deliveries.*
- **Worker**: A team member who executes *Tasks* during an *Iteration.* A *Worker* could have an associated *Role.*
- **Artifact**: *Artifacts* are software products produced and consumed by *Tasks.* It could include: *requirement specification*, *source code*, *test cases* and *management reports.* An *Artifact* could be associated with an *ArtifactDef*, and, in this case, the *Artifact* is an instance of an *ArtifactDef.*

Additionally, to comply with UML standards and improve the completeness of APMM, we have introduced the following OCL (OMG 2014 [199]) constraints as shown in Listing 5.2:

```
-- OCL constraints

constraints

context Project
    -- tasks in a case must also be in the project
  inv TasksAndCasesSameProject:
    self.case.task->forAll(t | self.task->includes(t))

    -- tasks in an iteration must also be in the project
  inv TasksAndIterationsSameProject:
    self.iteration.task->forAll(t | self.task->includes(t))

    -- the number of tasks in a project must
    -- be greater or equal to the number of iterations
  inv MoreTasksThanIterations:
    self.task->size >= self.iteration->size

    -- number of delivery intervals must be always lower than the number of deliveries
  inv MoreDeliveriesThanDeliveryIntervals:
    self.deliveryInterval->size > 0 implies
    self.deliveryInterval->size < self.delivery->size

    -- number of iteration intervals must be always lower than the number of iterations
  inv MoreIterationsThanIterationIntervals:
    self.iterationInterval->size > 0 implies
    self.iterationInterval->size < self.iteration->size
---
```

Listing 5.2: APMM constraints in OCL

- **TasksAndCasesSameProject**: defines that all tasks associated with a case must also be associated with the project associated with the case;

- **TasksAndIterationsSameProject**: defines that all tasks associated with an iteration must also be associated with the project associated with the iteration;
- **MoreTasksThanIterations**: defines that the number of tasks in a project must be greater or equal to the number of iterations;
- **MoreDeliveriesThanDeliveryIntervals**: defines that the number of deliveries in a project must be greater to the number of delivery intervals;
- **MoreIterationsThanIterationIntervals**: defines that the number of iterations in a project must be greater to the number of delivery intervals;

## 5.5 Conclusion

This chapter focused on the *Agile Project Metamodel*, which is another contribution of the presented research and that aids to achieve the goals G2 (*Define an agile smell representation approach*) and G3 (*Define a data extraction approach*).

This metamodel has an important role in the *AgileQube Approach* and it is directly used in the phases *Specification* and *Detection.* To define the proposed metamodel, we conducted a study divided into three phases. First, we defined the requirements for the metamodel (i.e. which elements the metamodel should provide to represent an agile project under the perspective of the agile smells presented in the *Catalogue of Agile Smells*). Second, we conducted a literature review to investigate how existing software development metamodels represent (or partially represent) the elements identified in the first phase. This part of the study revealed that there is no metamodel that can be used to fully represent the selected elements. Third, we proposed a metamodel, that we named *Agile Project Metamodel*, that fills this gap by combining elements from existing metamodel with the elements neglected by these metamodels. The proposed metamodel has a total of 13 elements and the 3 most important concepts are: *Project*, *Iteration*, and *Task.*

# Chapter 6

# *Agile Smell Schema*

*This chapter presents the Agile Smell Schema, a schema that is part of the AgileQube Approach and that enables the specification and configuration of agile smells. The chapter is organized in the following sections: Introduction, Schema Specification, Properties of the Agile Smell Schema, An example, The Top 10 Agile Smells Specifications, and Conclusion.*

## 6.1 Introduction

The systematic specification of an agile smell in the *AgileQube Approach* is driven by two components: *(a)* a metamodel to represent the data of the assessed agile project, and *(b)* a schema that is used to describe the structure of the agile smell. In this chapter, we present the *Agile Smell Schema*, a schema that, along with the *Agile Project Metamodel*, enables the systematic specification of the agile smells in the *AgileQube Approach* and, therefore, helps this research to fulfill the goal G2 (*Define an agile smell representation approach*).

As shown in Figure 6.1, the *Agile Smell Schema* has a key role in the *Specification*, *Configuration* and *Detection* phases. The schema is used by the *Programmer*, who translates textual descriptions of agile smells into systematic specifications. These specifications, after properly configured, are ultimately used by the *Detection Engine* to detect the agile smells.

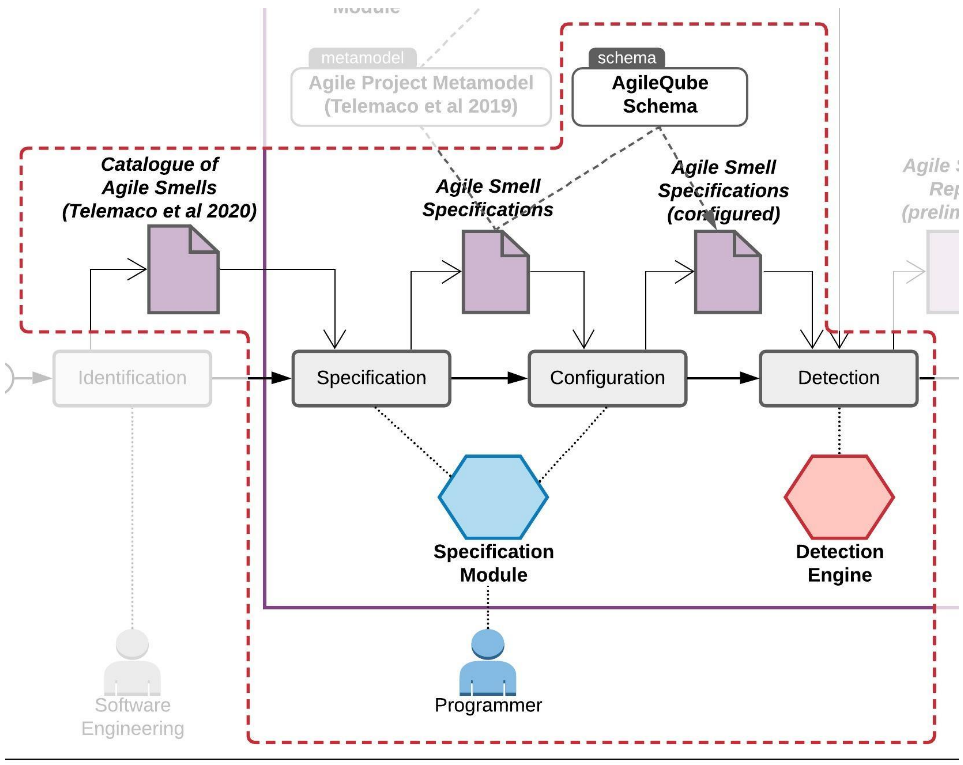

metamodel
Agile Project Metamodel (Telemaco et al 2019)
schema
AgileQube Schema
Catalogue of Agile Smells (Telemaco et al 2020)
Agile Smell Specifications
Agile Smell Specifications (configured)
Identification
Specification
Configuration
Detection
Specification Module
Detection Engine
Software Engineering
Programmer

Figure 6.1: *Agile Smell Schema* overview

## 6.2 Schema Specification

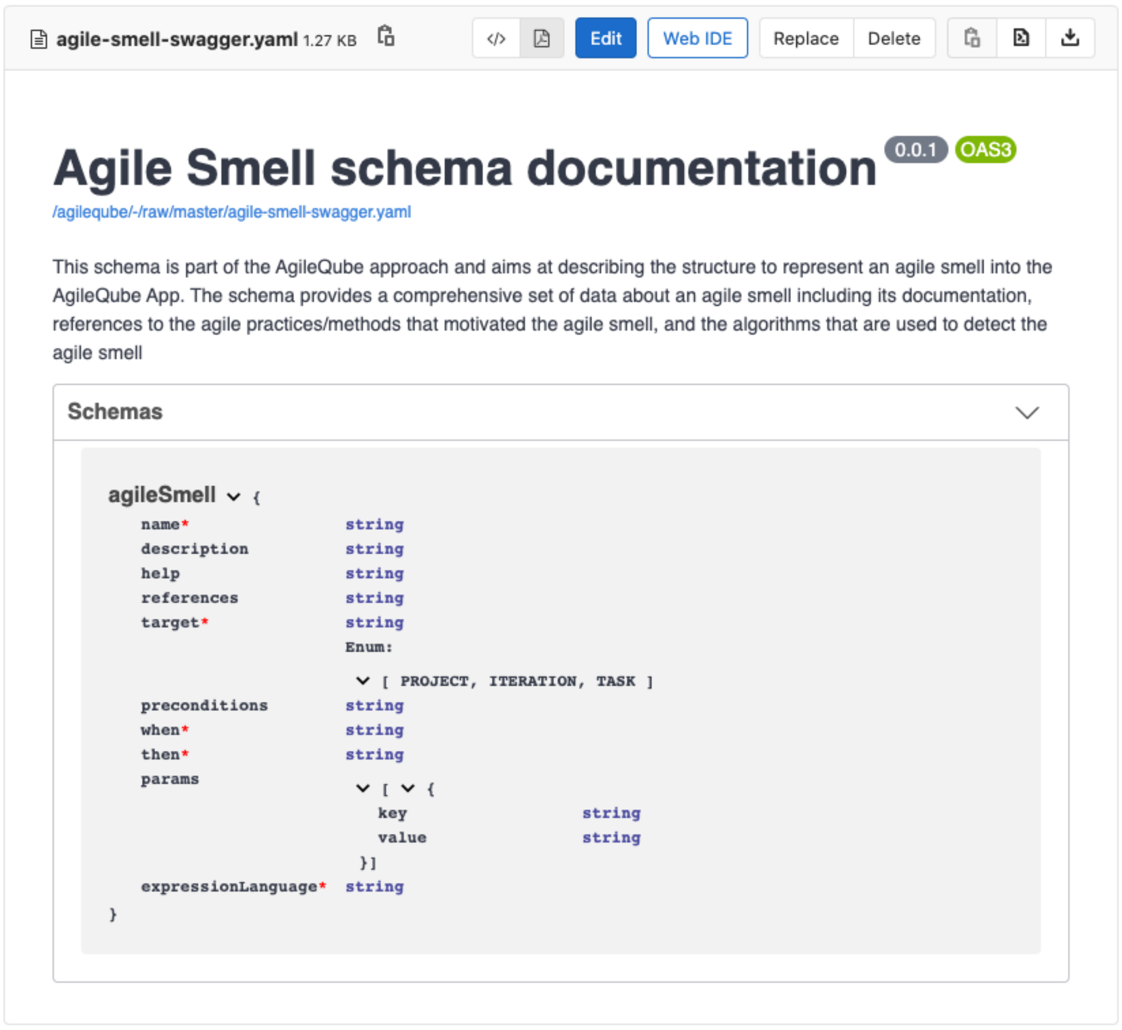
We are using the *Data Model* module of the OpenAPI 3.0 [1] to describe the *Agile Smell Schema*. The OpenAPI Specification (OAS) (FOUNDATION 2018 [200]), formerly known as Swagger Specification, is a standard used to describe, produce, consume, and visualize REST APIs. We selected OAS because it has gained considerable influence over the last years and has become one of the most popular standards for API/model documentation (KOREN and KLAMMA 2018 [201]). In addition to being a programming language-agnostic, stable and open specification supported by a large community and by companies such as Google, Microsoft, and IBM (SANDOVAL 2018 [202]). In the *Data Model* module of the OAS, the data types are based on a subset of the JSON Schema Specification [2] and described as *Schema* objects. Listing 6.1 presents the OAS specification of the *Agile Smell Schema*.

[1]https://swagger.io/docs/specification/data-models/
[2]https://tools.ietf.org/html/draft-wright-json-schema-00

```yaml
openapi: 3.0.0
info:
  title: Agile Smell Schema documentation
  description: This schema is part of the AgileQube approach and aims at describing the
    structure to represent an agile smell into the AgileQube App. The schema provides a
    comprehensive set of data about an agile smell including its documentation,
    references to the agile practices/methods that motivated the agile smell, and the
    algorithms that are used to detect the agile smell
  version: 0.0.1
components:
  schemas:
    agileSmell:
      type: object
      properties:
        name:
          type: string
        description:
          type: string
        help:
          type: string
        references:
          type: string
        target:
          type: string
          enum:
            - PROJECT
            - ITERATION
            - TASK
        preconditions:
          type: string
        when:
          type: string
        then:
          type: string
        params:
          type: array
          items:
            type: object
            properties:
              key:
                type: string
              value:
                type: string
        expressionLanguage:
          type: string
      required:
        - name
        - target
        - when
        - then
        - expressionLanguage
```

Listing 6.1: *Agile Smell Schema* OpenAPI specification

A user-friendly representation of the specification presented in Listing 6.1 is shown in Figure 6.2.



Agile Smell schema documentation

/agileqube/-/raw/master/agile-smell-swagger.yaml

This schema is part of the AgileQube approach and aims at describing the structure to represent an agile smell into the AgileQube App. The schema provides a comprehensive set of data about an agile smell including its documentation, references to the agile practices/methods that motivated the agile smell, and the algorithms that are used to detect the agile smell

```
agileSmell {
    name*                 string
    description           string
    help                  string
    references            string
    target*               string
                          Enum:
                           [ PROJECT, ITERATION, TASK ]
    preconditions         string
    when*                 string
    then*                 string
    params                 [ {
                            key                   string
                            value                 string
                           }]
    expressionLanguage*   string
}
```

Figure 6.2: *Agile Smell Schema* documentation

## 6.3 Properties of the *Agile Smell Schema*

The *Agile Smell Schema* provides to the approach a comprehensive set of data about an agile smell including its documentation, references to the agile practices/methods that motivated the agile smell, and the algorithms that are used to detect the agile smell. The schema has 10 properties:

1. *Name*;
2. *Description*;
3. *Help*;
4. *References*;
5. *Target*;
6. *Preconditions*;
7. *When*;
8. *Then*;
9. *Params*;
10. *Expression Language.*

### 6.3.1 *Name*, *Description*, *Help*, and *References*

The *Agile Smell Schema* has four properties to document the agile smell: *Name*, *Description*, *Help* and *References*.

The *Name* property is a short name of the agile smell. The *Description* property is a extended description of the agile smell. The *Help* property describes orientations to eliminate or avoid the agile smell. The *References* property indicates the references that support the agile smell.

### 6.3.2 *Target*

The *Target* property denotes the concept (or entity) in the *Agile Project Metamodel* that is the target of the agile smell. The properties (*Precondition*, *When* and *Then*) of the schema reference this concept in their expressions. The *Target* property can assume the following values: *Project*, *Iteration*, and *Task*.

### 6.3.3 *Preconditions*, *When*, *Then*, *Params* and *Expression Language*

The *Agile Smell Schema* has three expression properties: *Preconditions*, *When*, and *Then*. These properties denote the algorithms that are evaluated by the *Detection Engine* in the *Detection* phase.

The *Preconditions* property is an expression that verifies whether the preconditions to execute the other two expressions (*When* and *Then*) were met. It should check if the data required to identify the agile smell are available. For example, the *Precondition* expression of an agile smell related to task priority (eg *Lower Priority Tasks Executed First*) should verify if the information regarding the priority of the task is available. Otherwise, it will not be possible to identify the occurrence or absence of the agile smell in the task.

The *When* property is an expression that defines whether the agile smell should be evaluated or not. It verifies the conditions that define if the agile smell is applicable for the given target. For example, the *When* expression of an agile smell that is applicable only to finished iterations should verify if the iteration is finished.

The *Then* property is an expression that ultimately identifies the occurrence or absence of the agile smell. The agile smell is identified when the *Then* expression evaluates to `true`.

The *Expression Language* property defines the expression language used to specify the properties *Preconditions*, *When* and *Then*.

The *Params* property defines the parameters of an agile smell specification. A parameter is a value used in the expressions above presented (*Preconditions*, *When*, and *Then*) that may vary from company to company and therefore a constant value (or hard coded value) is not appropriated. It is possible to set a default value to a given parameter, but each company should calibrate the agile smell parameters to fit its context and processes. The team's expertise with agile methods and the characteristics of the organization's processes are some elements that influence the calibration of the agile smells parameters. For example, experienced teams can define more flexible values for the parameters while less experienced teams tend to use more restrictive values for their parameters. The *Params* property is not mandatory since an agile smell specification may not have parameters.

## 6.4 An example

Before diving into the specifications of the 10 agile smells presented in the *Catalogue of Agile Smells* discussed in Chapter 4, we will present an example that illustrates how a simple agile smell could be specified using the proposed data structure. We will take as example an agile smell that states that *all tasks in a given iteration should have their priority defined prior to the start of the iteration.* The specification of this agile smell, that we named *Task without priority in an open iteration*, is shown in Listing 6.2.

```
{
 name: { 'Task without priority in an open iteration' },
 description: {
  'All tasks in an open iteration should have a priority
  defined'
 },
 target: { TASK },
 help: {
  'To solve this agile smell, the development team should
  define the priority of all tasks before starting the
  iteration. If an iteration starts and there are tasks
  without a priority defined, the development team should
  define the priority of those tasks as soon as possible.'
 },
 preconditions: {
  boolean isAssociatedToAnIteration = task.iteration
  boolean iterationHasStatus = task.iteration.status
  return isAssociatedToAnIteration && iterationHasStatus
 },
 when: {
  boolean isOpen = task.iteration.status == 'open'
  return isOpen
 },
 then: {
  boolean hasPriority = task.priority
  return hasPriority
 },
 expressionLanguage: { Groovy }
}
```

Listing 6.2: An example of an agile smell specification

The properties *name*, *description* and *help* are text-based elements used to document the agile smell. The *name* property provides the unique identifier to the agile smell: *Task without priority in a open iteration*. The *description* property gives a detailed explanation of the agile smell including a brief mention of the agile practices that motivated it. The *help* property provides instructions for avoiding and fixing the agile smell. The *target* property is defined to `Task` which means the agile smell is related to the *Task* concept of the *Agile Project Metamodel*. The *preconditions* property is an expression that evaluates two conditions: *(a)* if the task has an associated iteration and *(b)* if the associated iteration has a defined status. The preconditions for this agile smell are only met when the two conditions are `true`. This means that if a task is not associated with an iteration or if the associated iteration has no defined status, the preconditions to identify this agile smell on that task are not met. The *when* property is an expression that checks if the iteration associated to the assessed task is open. This means the agile smell is applicable only when the iteration associated to the task is open. The *then* property is an expression that checks whether the given task has a defined priority. When this expression evaluates to `true`, the agile smell is identified. Finally, the *expression-language* property sets *Groovy* (KOENIG *et al.* 2007 [203]) as the expression language for the previous expressions (*preconditions*, *when*, and *then*).

## 6.5 The Top 10 Agile Smells Specifications

In this section, we discuss the specifications of the agile smells from the *Catalogue of Agile Smells* presented in Chapter 4. The *description* and *help* properties were omitted to improve the readability of the specifications. However, the absence of these properties does not jeopardize the understanding of the specification or the detection of the agile smell by the *Detection Engine*.

### 6.5.1 *Lower Priority Tasks Executed First* specification

The specification of the agile smell *Lower Priority Tasks Executed First* is shown in Listing 6.3. The *target* property is defined to *Iteration* indicating the agile smell is applied to the iterations of an agile project. The *precondition* expression checks if the iteration has a status and tasks associated. The *when* expression checks if the iteration is open and has *to do* and *doing* tasks. The *then* expression finds the *to do* task with the highest priority and calculates the lists of *doing* tasks. The agile smell is detected when the algorithm verifies that there is a *doing* task with a lower priority than the highest priority *to do* task.

```
{
 name: { 'AS01 Lower Priority Task Executed First' },
 target: { ITERATION },
 preconditions: {
  boolean hasStatus = iteration.status
  boolean hasTasks = iteration.tasks
  return hasStatus && hasTasks
 },
 when: {
  def isOpen = iteration.status == 'open'
  def todoTasks = iteration.tasks.findAll { it.status == 'todo' && it.priority }
  def doingTasks = iteration.tasks.findAll { it.status == 'doing' && it.priority }

  boolean hasDoingTasks = !doingTasks.isEmpty()
  boolean hasTodoTasks = !todoTasks.isEmpty()
  return isOpen && hasDoingTasks && hasTodoTasks
 },
 then: {
  def highestPriorTodoTask = iteration.tasks.findAll {
   it.status == 'todo' && it.priority
  }?.max {
   it.priority?.value
  }?.priority?.value
  def doingTasks = iteration.tasks.findAll {
   it.status == 'doing' && it.priority
  }
  boolean lowerPriorityTaskExecutedFirst = false
  doingTasks.each { doingTask ->
   if (doingTask.priority.value < highestPriorTodoTask) {
    lowerPriorityTaskExecutedFirst = true
   }
  }
  return lowerPriorityTaskExecutedFirst
 },
 expressionLanguage: { Groovy }
}
```

Listing 6.3: *Lower Priority Tasks Executed First* specification

### 6.5.2 *Absence of Frequent Deliveries* specification

The specification of the agile smell *Absence of Frequent Deliveries* is presented in Listing 6.4. The *target* property is defined to *Project*. The *preconditions* expression defines the project has to have two collections associated: delivery intervals and iterations. The *when* expression defines this agile smell is applied only to projects that have more than 1 iteration. The *then* expression verifies whether there are delivery intervals longer than the maximum allowed delivery interval. The *params* property defines a parameter to indicate the maximum allowed delivery interval.

```
{
 name: { 'AS02 Not Frequent Deliveries' },
 target: { PROJECT },
 preconditions: {
  boolean hasDeliveryIntervals = project.deliveryIntervals
  boolean hasIterations = project.iterations
  return hasDeliveryIntervals && hasIterations
 },
 when: {
  boolean hasMoreThan1It = project.iterations.size() > 1
  return hasMoreThan1It
 },
 then: {
  def notCompliantDelivIntervals = project.deliveryIntervals.findAll {
    it.duration.inDays() > param.maxDeliveryInterval
  }
  return !notCompliantDelivIntervals.isEmpty()
 },
 params: {
  maxDeliveryInterval: 15
 },
 expressionLanguage: { Groovy }
}
```

Listing 6.4: *Absence of Frequent Deliveries* specification

### 6.5.3 *Iteration Without a Deliverable* specification

The specification of the agile smell *Iteration Without a Deliverable* is shown in Listing 6.5. The *target* property is defined to *Iteration*. The *precondition* expression specifies the iteration has to have associated tasks. The *when* expression indicates the agile smell is always applied regardless of the context of the iteration. The *then* expression finds deploy tasks in the iteration and the agile smell is detected when there is no deploy task in the iteration. The last element, *params*, defines a parameter to specify the name of the deploy task.

```
{
 name: { 'AS03 Iteration Without A Deliverable' },
 target: { ITERATION },
 preconditions: {
  boolean hasTasks = iteration.tasks
  return hasTasks
 },
 when: {
  boolean always = true
  return always
 },
 then: {
  def deployTasks = iteration.tasks.findAll {
   task -> task.implementedActivities.findAll{
    activity -> activity.name == params.deployActivityName
   }
  }

  boolean doesNotHaveADeployTask = deployTasks.isEmpty()
  return doesNotHaveADeployTask
 },
 params: {
  deployActivityName: 'Deploy New Version'
 },
 expressionLanguage: { Groovy }
}
```

Listing 6.5: *Iteration Without a Deliverable* specification

### 6.5.4 *Goals Not Defined or Poorly Defined* specification

The specification of the agile smell *Goals Not Defined or Poorly Defined* is shown in Listing 6.6. The *target* property is defined to *Iteration*. The absence of the *precondition* property means there is no precondition to this agile smell. The *when* expression indicates the agile smell is always applied regardless of the context of the iteration. The *then* expression assesses the iteration description and the agile smell is detected when the iteration has no description or when the description does not have the minimum number of characters required. The *params* property defines a parameter to specify the minimum length of an iteration description.

```
{
 name: { 'AS04 Goals Not Defined or Poorly Defined' },
 target: { ITERATION },
 when: {
  boolean always = true
  return always
 },
 then: {
  if (!iteration.description) {
   boolean doesNotHaveAGoal = true
   return doesNotHaveAGoal
  }

  def itDescLen = iteration.description.length()
  def minDescLen = params.minimalDescriptionLength
  boolean doesNotHaveAClearGoal = itDescLen <= minDescLen
  return doesNotHaveAClearGoal
 },
 params: {
  minimalDescriptionLength: 50
 },
 expressionLanguage: { Groovy }
}
```

Listing 6.6: *Goals Not Defined or Poorly Defined* specification

### 6.5.5 *Iteration Without an Iteration Planning* specification

The specification of the agile smell *Iteration Without an Iteration Planning* is shown in Listing 6.7. The *target* property is defined to *Iteration*. The *precondition* expression specifies the iteration has to have a status and a list of associated tasks. The *when* expression indicates the agile smell is applied only to open iterations. The *then* expression finds planning tasks in the iteration and the agile smell is detected when there is no planning task in the iteration. The last element, *params*, defines a parameter to specify the name of the planning task.

```
{
 name: { 'AS05 Iteration Without An Iteration Planning' },
 target: { ITERATION },
 preconditions: {
  boolean hasStatus = iteration.status
  boolean hasTasks  = iteration.tasks
  return hasStatus && hasTasks
 },
 when: {
  def isOpen = iteration.status == 'open'
  return isOpen
 },
 then: {
  def planningTasks = iteration.tasks.findAll {
     task -> task.implementedActivities.findAll {
        activity -> activity.name == params.planningActivityName
     }
  }

  boolean doesNotHaveAPlanTask = planningTasks.isEmpty()
  return doesNotHaveAPlanTask
 },
 params: {
  planningActivityName: 'Iteration Planning'
 },
 expressionLanguage: { Groovy }
}
```

Listing 6.7: *Iteration Without an Iteration Planning* specification

### 6.5.6 *Complex Tasks* specification

Listing 6.8 presents the specification of the agile smell *Complex Tasks*. The *target* property is defined to *Task* indicating the agile smell is applied to all tasks in the agile project. The *precondition* expression indicates that the task should have a defined effort. The *when* expression defines that the agile smell is always applied regardless of the context of the task. The *then* expression assesses the task effort and the agile smell is detected when the task effort is greater than the maximum allowed effort. The *params* property defines a parameter to indicates the maximum allowed task effort.

```
{
 name: { 'AS06 Complex Task' },
 target: { TASK },
 preconditions: {
  boolean hasEstimatedEffort = task.effort?.estimated
  return hasEstimatedEffort
 }
 when: {
  boolean always = true
  return always
 },
 then: {
  def taskEffort = task.effort.estimated
  def maxAllowedEffort = params.maxAllowedEffort
  boolean isAComplexTask = taskEffor > maxAllowedEffort
  return isAComplexTask
 },
 params: {
  maxAllowedEffort: 8
 },
 expressionLanguage: { Groovy }
}
```

Listing 6.8: *Complex Tasks* specification

### 6.5.7 *Iteration Without an Iteration Retrospective* specification

The specification of the agile smell *Iteration Without an Iteration Retrospective* is presented in Listing 6.9. The *target* property is defined to *Iteration*. The *precondition* expression specifies the iteration has to have a status and a list of associated tasks. The *when* expression indicates the agile smell is applied only to closed iterations. The *then* expression finds retrospective tasks in the iteration and the agile smell is detected when there is no retrospective task in the iteration. The last element, *params*, defines a parameter to specify the name of the retrospective task.

```
{
 name: { 'AS07 Iteration Without An Iteration Retrospective' },
 target: { ITERATION },
 preconditions: {
  boolean hasStatus = iteration.status
  boolean hasTasks  = iteration.tasks
  return hasStatus && hasTasks
 },
 when: {
  boolean isClosed = iteration.status == 'closed'
  return isClosed
 },
 then: {
  def retrospectiveTasks = iteration.tasks.findAll {
      task -> task.implementedActivities.findAll {
        activity -> activity.name == params.retrospectActivityName
      }
  }

  boolean doesNotHaveARetrospetiveTask = retrospectiveTasks.isEmpty()
  return doesNotHaveARetrospetiveTask
 },
 params: {
  retrospectActivityName: 'Iteration Retrospective'
 }
 expressionLanguage: { Groovy }
}
```

Listing 6.9: *Iteration Without an Iteration Retrospective* specification

### 6.5.8 *Absence of Timeboxed Iteration* specification

Listing 6.10 presents the specification of the agile smell *Absence of Timeboxed Iteration*. The *target* property is defined to *Iteration*. The *preconditions* property specifies the iteration has to have an estimated and a real finish dates. The *when* expression defines the agile smell is only applied to closed iteration. The *then* expression calculates the difference between the estimated and real fish dates and the agile smell is detected when this difference is greater than the maximum difference allowed. The last element, *params*, defines a parameter to indicate the maximum allowed difference between the estimated and the real finish dates.

```
{
 name: { 'AS08 Not Timeboxed Iterations' },
 target: { ITERATION },
 preconditions: {
  boolean hasFinishDate = iteration.finishDate
  boolean hasEstimatedFinishDate = iteration.estimatedFinishDate
  return hasFinishDate && hasEstimatedFinishDate
 }
 when: {
  boolean isClosed = iteration.status == 'closed'
  return isClosed
 },
 then: {
  def finishDate = iteration.finishDate
  def estimatedFinishDate = iteration.estimatedFinishDate
  def diff = finishDate.minus(estimatedFinishDate)
  boolean notTimeboxed = diff <= params.maxFinishVariation
  return notTimeboxed
 },
 params: {
  maxFinishVariation: 3
 }
 expressionLanguage: { Groovy }
}
```

Listing 6.10: *Absence of Timeboxed Iteration* specification

### 6.5.9 *Iteration Started without an Estimated Effort* specification

The specification of the agile smell *Iteration Started without an Estimated Effort* is shown in Listing 6.11. The *target* property is defined to *Iteration*. The *precondition* expression specifies the iteration has to have a status and a list of associated tasks. The *when* expression indicates the agile smell is only applied to open iterations. The *then* expression assesses the tasks associated with the iteration and the agile smell is detected when there is at least one task without an estimated effort.

```
{
 name: { 'AS09 Iteration Started Without Estimated Effort' },
 target: { ITERATION },
 preconditions: {
  boolean hasStatus = iteration.status
  boolean hasTasks = !iteration.tasks.isEmpty()
  return hasStatus && hasTasks
 },
 when: {
  boolean isOpen = iteration.status == 'open'
  return isOpen
 },
 then: {
  def notEstTasks = iteration.tasks.findAll {
   !it.effort || !it.effort.estimated
  }

  boolean notAllTasksEstimated = !notEstTasks.isEmpty()
  return notAllTasksEstimated
 },
 expressionLanguage: { Groovy }
}
```

Listing 6.11: *Iteration Started without an Estimated Effort* specification

### 6.5.10 *Iteration Without an Iteration Review* specification

The specification of the agile smell *Iteration Without an Iteration Review* is shown in Listing 6.12. The *target* property is defined to *Iteration*. The *precondition* expression specifies the iteration has to have a status and a list of associated tasks. The *when* expression indicates the agile smell is applied only to closed iterations. The *then* expression finds review tasks in the iteration and the agile smell is detected when there is no review task in the iteration. The last element, *params*, defines a parameter to specify the name of the review task.

```
{
 name: { 'AS10 Iteration Without An Iteration Review' },
 target: { ITERATION },
 when: {
  boolean isClosed = iteration.status == 'closed'
  return isClosed
 },
 then: {
  def reviewTasks = iteration.tasks.findAll {
   task -> task.implementedActivities.findAll{
    activity -> activity.name == params.reviewActivity
   }
  }

  boolean doesNotHaveAReviewTask = reviewTasks.isEmpty()
  return doesNotHaveAReviewTask
 },
 params: {
  reviewActivity: 'Iteration Review'
 }
 expressionLanguage: { Groovy }
}
```

Listing 6.12: *Iteration Without an Iteration Review* specification

## 6.6 Conclusion

This chapter focused on the *Agile Smell Schema*, one of the components of the *AgileQube Approach* that enables the specification of agile smells into the approach. This contribution, along with the *Agile Project Metamodel*, aids this research to achieve the goal G2 (*Define an agile smell representation approach*). This schema is used by a *Programmer*, who translates the textual descriptions of the agile smells into specifications during the *Specification* and *Configuration* phases. We presented the specification of the schema using the *Data Model* module of OpenAPI 3.0. The *Agile Smell Schema* has 10 properties (*Name*, *Description*, *Help*, *References*, *Target*, *Preconditions*, *When*, *Then*, and *Expression Language*) that provide a comprehensive set of information about the agile smell, including its documentation, references to agile practices and agile methods related to the agile smell, and the algorithms that are ultimately used by the *Detection Engine* to detect the agile smell. We presented an example to illustrate how to use the schema to specify an agile smell as well as the specification of 10 agile smells from the *Catalogue of Agile Smells* using the *Agile Smell Schema*.

# Chapter 7

# Supporting Infrastructure: ***AgileQube App***

*This chapter presents a supporting infrastructure for the AgileQube Approach and is organized in the following sections: Introduction, AgileQube App Architecture, AgileQube App Components, and Conclusion.*

## 7.1 Introduction

The *AgileQube App* is the one of the contributions of this research and it aims to answer the goal G4 (*Define and implement a computational system*). We believe that developing such a infrastructure to support AA built on top of a solid conceptual foundation such as ours, is a valuable contribution, not only to test our ideas but also to motivate other researchers and practitioners in pursuing AA in their projects. The *AgileQube App* supports three phases of the *AgileQube Approach* (*Specification*, *Configuration*, *Detection*, and *Validation*) and has four components (*Specification Module*, *ETL Module*, *Detection Engine*, and *Validation Module*) as depicted in Figure 7.1.

Two key users interact with the *AgileQube App*: a *Programmer* that uses the *Specification Module* in the phases *Specification* and *Configuration* and a *Team Member* that performs the phase *Validation* through the *Validation Module.*

## 7.2 ***AgileQube App*** Architecture

The *AgileQube App* is a web based application built according to the software architecture shown in Figure 7.2. The architecture is divided into three layers: front ent,

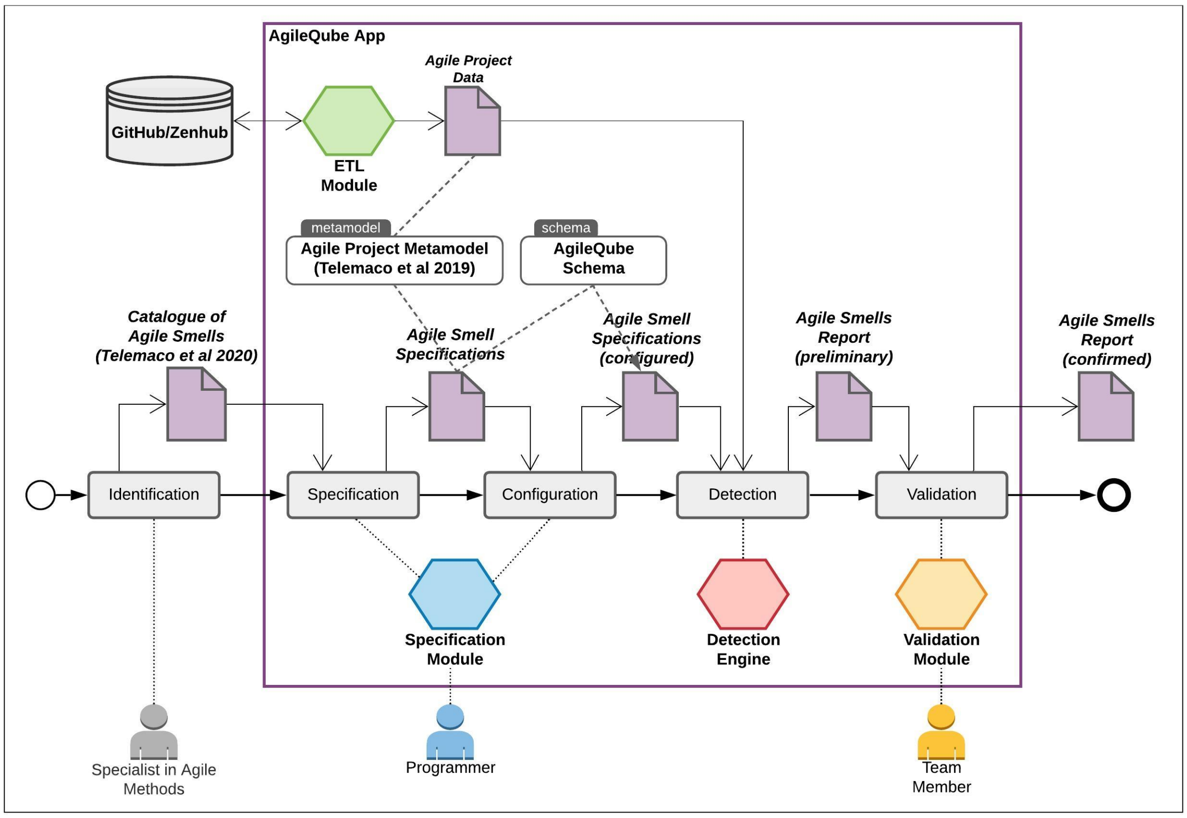

AgileQube App
Agile Project Data
GitHub/Zenhub
ETL Module
metamodel
Agile Project Metamodel (Telemaco et al 2019)
schema
AgileQube Schema
Catalogue of Agile Smells (Telemaco et al 2020)
Agile Smell Specifications
Agile Smell Specifications (configured)
Agile Smells Report (preliminary)
Agile Smells Report (confirmed)
Identification
Specification
Configuration
Detection
Validation
Specification Module
Detection Engine
Validation Module
Specialist in Agile Methods
Programmer
Team Member

Figure 7.1: AgileQube App overview

back end, and database. The front end layer is based on the framework *Vue.js*$^{TM}$[1] which is an open source JavaScript framework focused on developing single page applications (SPA) (MIKOWSKI and POWELL 2013 [204]). The back end layer is based on the framework *Spring*$^{TM}$[2] that is an application framework and inversion of control container for the Java platform focused on creating stand-alone and preconfigured applications. The database layer is an instance of a *PostgreSQL*$^{TM}$[3] which is a free and open-source relational database management system (RDBMS).

## 7.3 *AgileQube App* Components

This section presents the main components that compose the *AgileQube App*.

### 7.3.1 *Specification Module*

As highlighted in Figure 7.3, the *Specification Module* aids the *Programmer* in the execution of the *Specification* and *Configuration* phases.

[1]https://vuejs.org/
[2]https://spring.io/
[3]https://www.postgresql.org/

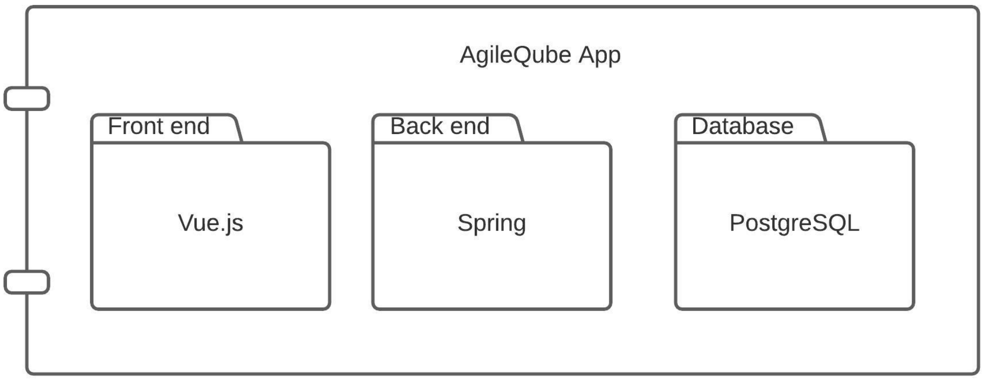
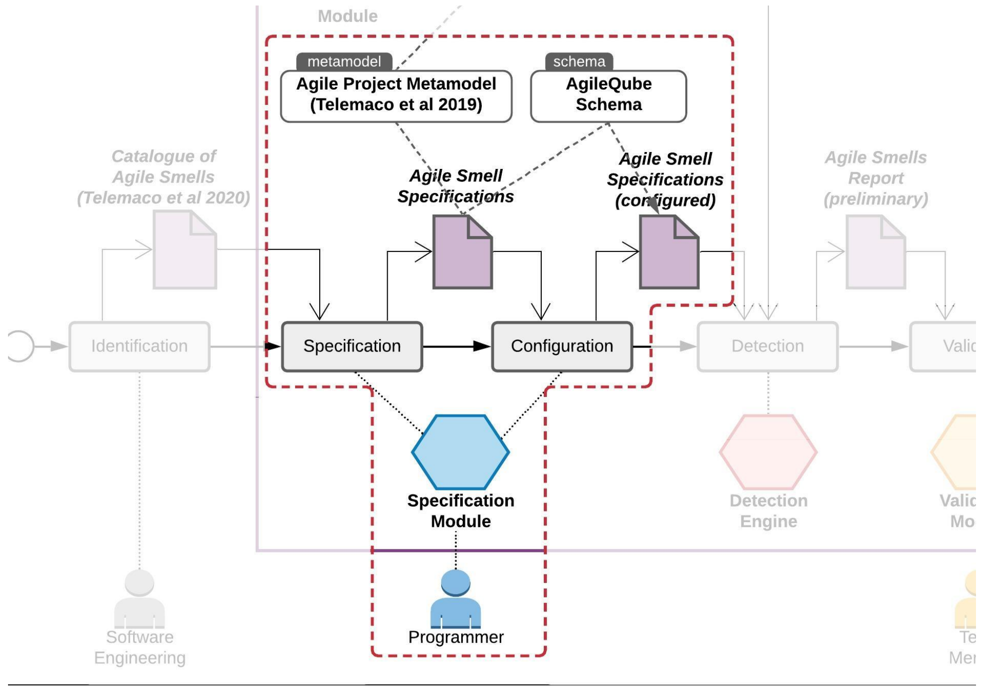


Figure 7.2: *AgileQube App* Architecture overview



Module
metamodel
Agile Project Metamodel (Telemaco et al 2019)
schema
AgileQube Schema
Catalogue of Agile Smells (Telemaco et al 2020)
Agile Smell Specifications
Agile Smell Specifications (configured)
Agile Smells Report (preliminary)
Identification
Specification
Configuration
Detection
Specification Module
Detection Engine
Software Engineering
Programmer

Figure 7.3: *Specification Module* overview

This module provides the features that enable a *Programmer* to specify and configure the agile smells using the *Agile Smell Schema*. Figures 7.4 and 7.5 show the interfaces that compose the *Specification Module*. The agile smell specification form (Figure 7.4) provides the following input entries that correspond to the properties in the *Agile Smell Schema*: *name*, *description*, *target*, *preconditions*, *when*, *then*, and *params*. The list of the agile smells specified and configured in the *AgileQube App* is shown in Figure 7.5.

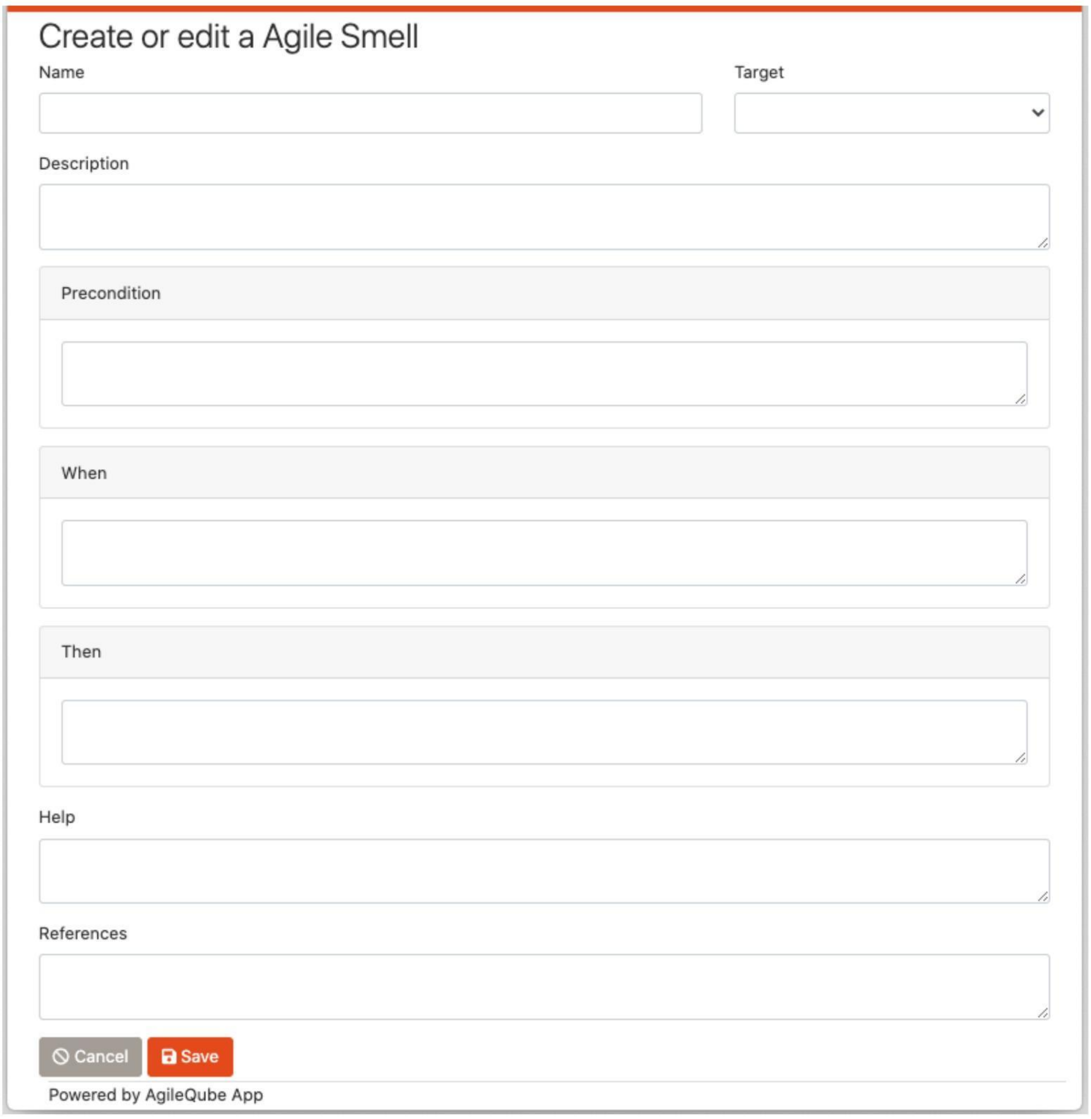


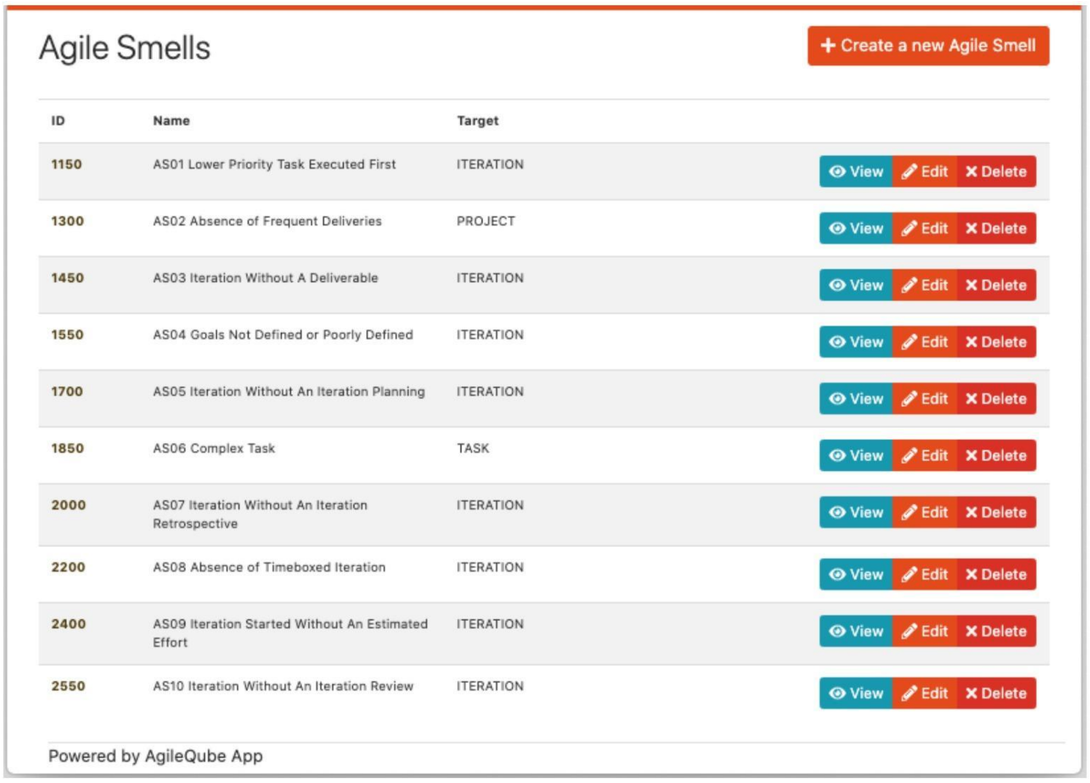
Figure 7.4: Agile smell specification form

### 7.3.2 *ETL Module*

ETL (Extracting, Transforming and Loading) is the general procedure of transfer data from one or more sources to a destination system which may represent the data differently from the source(s) or in a different context than the source(s) (VASSILIADIS *et al.* 2002 [205]).

As depicted in Figure 7.6, the *ETL Module* in the *AgileQube App* is responsible for: *(a)* extracting the data related to the assessed agile project from a project management systems such as *ZenHub*[4]; *(b)* transforming the extracted data into the format supported by the *Detection Engine* (which is the *Agile Project Metamodel*); and *(c)* inputting the transformed data into the *Detection Engine*.

[4]`https://www.zenhub.com/`



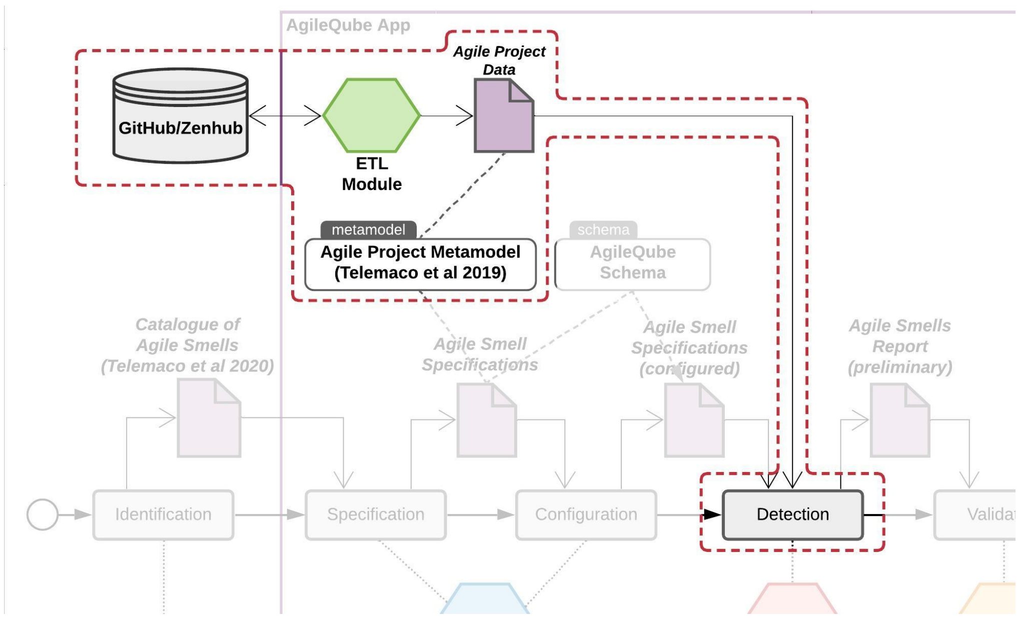
Agile Smells

+ Create a new Agile Smell

| ID | Name | Target | |
|---|---|---|---|
| 1150 | AS01 Lower Priority Task Executed First | ITERATION | View Edit Delete |
| 1300 | AS02 Absence of Frequent Deliveries | PROJECT | View Edit Delete |
| 1450 | AS03 Iteration Without A Deliverable | ITERATION | View Edit Delete |
| 1550 | AS04 Goals Not Defined or Poorly Defined | ITERATION | View Edit Delete |
| 1700 | AS05 Iteration Without An Iteration Planning | ITERATION | View Edit Delete |
| 1850 | AS06 Complex Task | TASK | View Edit Delete |
| 2000 | AS07 Iteration Without An Iteration Retrospective | ITERATION | View Edit Delete |
| 2200 | AS08 Absence of Timeboxed Iteration | ITERATION | View Edit Delete |
| 2400 | AS09 Iteration Started Without An Estimated Effort | ITERATION | View Edit Delete |
| 2550 | AS10 Iteration Without An Iteration Review | ITERATION | View Edit Delete |

Powered by AgileQube App

Figure 7.5: Agile smells registered in the system



AgileQube App
Agile Project Data
GitHub/Zenhub
ETL Module
metamodel
Agile Project Metamodel (Telemaco et al 2019)
schema
AgileQube Schema
Catalogue of Agile Smells (Telemaco et al 2020)
Agile Smell Specifications
Agile Smell Specifications (configured)
Agile Smells Report (preliminary)
Identification
Specification
Configuration
Detection
Valida

Figure 7.6: *ETL Module* overview

Since there is a vast number of Software Project Management Tools in the literature (SAJAD *et al.* 2016 [206]), the *ETL Module* has an important role in the *AgileQube Approach* by isolating the complexity of dealing with the extraction of data from different sources with different data representation formats. Hence, this module was designed to load data from a wide range of tools such as *ZenHub*[4], HuBoard[5], Waffle.io[6], Redmine[7], BaseCamp[8], and Jira[9]. The implementation of the *ETL Module* presented in this study supports the integration with the *ZenHub* platform. This module is published as a separated and open-source project, the *Prisma ZenHub ETL*[10].

### Project Management on *GitHub*

*GitHub* is the most popular web-based git repository platform for software development. The platform has more than 30 million active users and hosts more than 100 million repositories (WARNER 2018 [207]). Besides the git repository (that is the main feature and that gives the name for the platform), *GitHub* offers many other native features such as: (a) Issue Tracker, (b) Web IDE, (c) Discussion Forum, and (d) Wiki (BLEIEL 2016 [208]). The platform is also highly extensible and provides a framework to aid the implementation of *GitHub* extensions. The *GitHub* marketplace[11] has more than 3000 plugins and applications distributed over categories such as: Project management, Chat, Code quality, Code review, Continuous integration, and IDEs.

Project management on *GitHub* is usually achieved with the use of the *Issue Tracker* feature (BLEIEL 2016 [208], ARORA *et al.* 2017 [209]) where an issue on *GitHub* denotes a task in the project. A *GitHub* issue has a very simple structure: *name*, *description*, and *labels*. Each issue can be associated with a milestone, which is an easy way to schedule a deadline for when work should be completed by. Issues can also be assigned to a developer (*GitHub* user).

### Project Management on *ZenHub*

*ZenHub* is an extension for *GitHub* that enhances the project management capability and fills some gaps present in the *GitHub* Issue Tracker (RACASAN 2020 [210]). In addition to the properties inherited from *GitHub* issue (*name*, *description*, *labels*, *milestone*, and *assignee*), a *ZenHub* issue has properties to represent information such as:

1. the estimated effort to complete the issue;

[5]`https://huboard.com/`
[6]`https://waffle.io//`
[7]`https://www.redmine.org/`
[8]`https://basecamp.com/`
[9]`https://www.atlassian.com/software/jira`
[10]`https://github.com/utelemaco/prisma-zenhub-etl/`
[11]`https://github.com/marketplace`

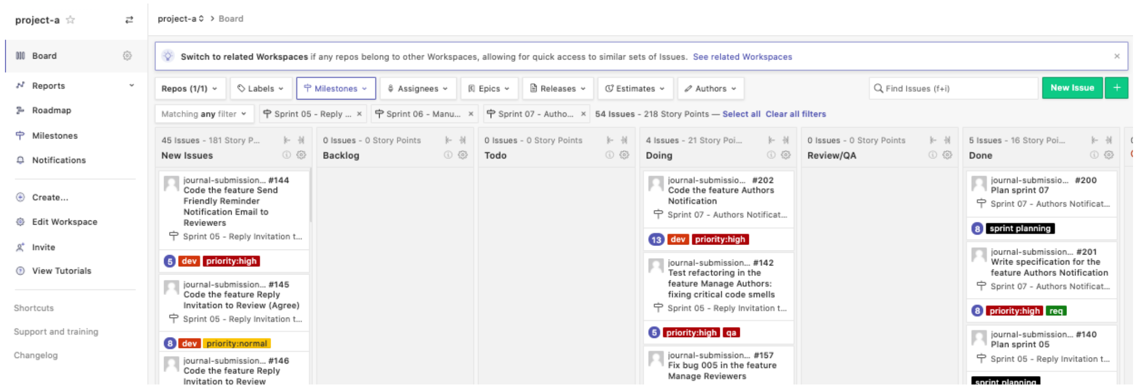
Table 7.1: Mapping between the *ZenHub*/*GitHub* platforms and the *Agile Project Metamodel*

| *GitHub* | *ZenHub* | *Agile Project Metamodel* |
|---|---|---|
| issue | issue | task |
| issue.name | issue.name | task.name |
| issue's labels | issue's labels | task.priority |
| issue's labels | issue.estimate | task.effort |
| milestone | milestone | iteration |

2. the current state of the issue in the workflow; and

3. dependencies between the issues.

Additionally, the *ZenHub* platform provides a customizable dashboard that gives an easy and comprehensive visualization of the issues workflow. Figure 7.7 shows the dashboard of a project hosted on *ZenHub*.



Switch to related Workspaces if any repos belong to other Workspaces, allowing for quick access to similar sets of Issues. See related Workspaces

54 Issues - 218 Story Points

45 Issues - 181 Story P... New Issues
- journal-submission... #144 Code the feature Send Friendly Reminder Notification Email to Reviewers — Sprint 05 - Reply Invitation t... — 5 dev priority:high
- journal-submission... #145 Code the feature Reply Invitation to Review (Agree) — Sprint 05 - Reply Invitation t... — 8 dev priority:normal
- journal-submission... #146 Code the feature Reply Invitation to Review

0 Issues - 0 Story Points Backlog

0 Issues - 0 Story Points Todo

4 Issues - 21 Story Poi... Doing
- journal-submissio... #202 Code the feature Authors Notification — Sprint 07 - Authors Notificat... — 13 dev priority:high
- journal-submission... #142 Test refactoring in the feature Manage Authors: fixing critical code smells — Sprint 05 - Reply Invitation t... — 5 priority:high qa
- journal-submission... #157 Fix bug 005 in the feature Manage Reviewers

0 Issues - 0 Story Points Review/QA

5 Issues - 16 Story Poi... Done
- journal-submissio... #200 Plan sprint 07 — Sprint 07 - Authors Notificat... — 8 sprint planning
- journal-submission... #201 Write specification for the feature Authors Notification — Sprint 07 - Authors Notificat... — 8 priority:high req
- journal-submission... #140 Plan sprint 05 — Sprint 05 - Reply Invitation t... — sprint planning

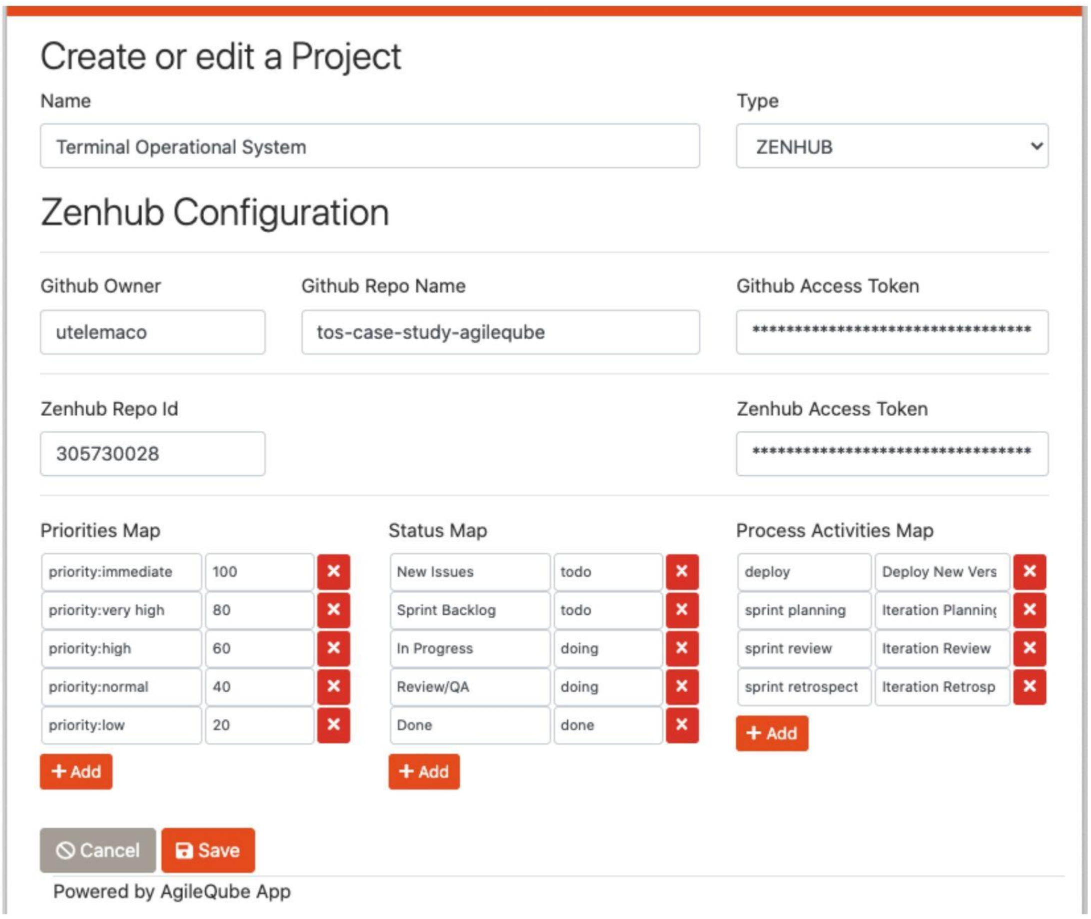
Figure 7.7: *ZenHub* dashboard - an example

### ***ZenHub*/*GitHub* to *Agile Project Metamodel***

We use the *issue* and *milestone* elements to represent, respectively, the *task* and *iteration* elements. Table 7.1 shows the mapping between *GitHub*/*ZenHub* elements and *Agile Project Metamodel* elements.

Note that *ZenHub* and *GitHub* do not support the inclusion of custom properties in their issue element. Instead of that, the platforms support the use of labels as a way to customize project specific information. Thus, we use labels for mapping some properties as detailed in Table 7.1. The priority of a task, for example, is represented by a label that may assume values such as `LOW`, `NORMAL`, and `HIGH`.

The configuration form that indicates to the ETL module how to load and transform data from *ZenHub*/*GitHub* is presented in Figure 7.8.


Create or edit a Project

Name: Terminal Operational System

Type: ZENHUB

Zenhub Configuration

Github Owner: utelemaco

Github Repo Name: tos-case-study-agileqube

Github Access Token: **********************************

Zenhub Repo Id: 305730028

Zenhub Access Token: **********************************

Priorities Map

| | |
|---|---|
| priority:immediate | 100 |
| priority:very high | 80 |
| priority:high | 60 |
| priority:normal | 40 |
| priority:low | 20 |

Status Map

| | |
|---|---|
| New Issues | todo |
| Sprint Backlog | todo |
| In Progress | doing |
| Review/QA | doing |
| Done | done |

Process Activities Map

| | |
|---|---|
| deploy | Deploy New Vers |
| sprint planning | Iteration Planning |
| sprint review | Iteration Review |
| sprint retrospect | Iteration Retrosp |

Figure 7.8: *ETL Module* form configuration for the *ZenHub* platform

### 7.3.3 *Detection Engine*

The *Detection Engine* appears as the core of the *AgileQube App* and has the responsibility of executing the *Detection* phase proposed in the *AgileQube Approach* that aims at identifying agile smells in a given agile project.

As depicted in Figure 7.9, the *Detection Engine* performs the *Detection* phase receiving two inputs (*(a)* an instance of the *Agile Project Metamodel* that represents the data from the assessed agile project and *(b)* a set of agile smell specifications coded in the *Agile Smell Schema*) and producing an *Agile Smells Report (preliminary)*.

During the *Detection* phase, each target element (the project, the iterations and the tasks) are assessed and for each agile smell specification, the *Detection Engine* calculates one of the following results:

1. **Preconditions not met**: this result indicates the preconditions to identify the presence of the agile smell were not met and therefore it is not possible to assert the presence or absence of the agile smell in the target element;

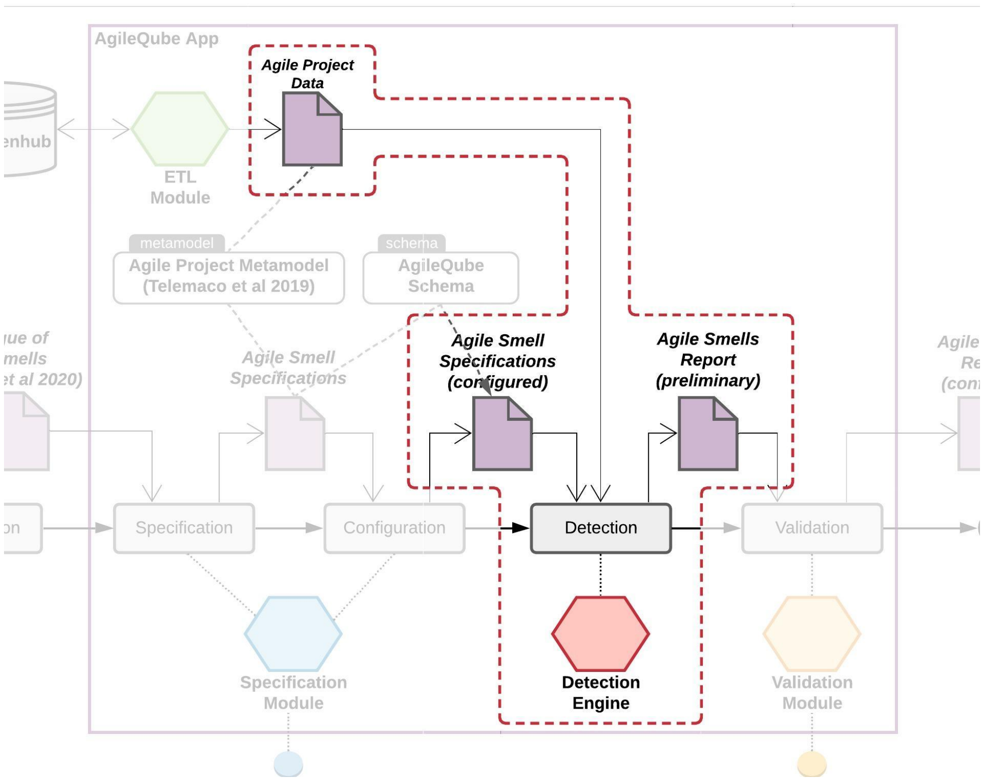


Figure 7.9: *Detection Engine* overview

2. **Not applicable**: the agile smell is not applicable to the target element;
3. **OK**: the agile smell was not detected in the target element; and
4. **Not OK**: the agile smell was detected in the target element.

### Expressions Evaluation

The *Detection Engine* evaluates the expressions *preconditions*, *when* and *then* from an agile smell specification to detect the occurrence of such agile smell as illustrated in Figure 7.10.

If the *preconditions* expression evaluates to `false`, the algorithm finishes and returns *Preconditions not met*. Otherwise, the *when* expression is evaluated. Thus, the *when* expression is only evaluated when the preconditions for evaluation are met. If the *when* expression evaluates to `false`, the algorithm finishes and returns *Not applicable*. Otherwise, the *then* expression is evaluated. Thus, the *then* expression is only evaluated when the previous expressions (*preconditions* and *when*) evaluate to `true`. When the *then* expression evaluates to `false`, the agile smell is not detected and the algorithm returns *OK*. Otherwise, when the *then* expression evaluates to `true`, the agile smell is detected and the algorithm returns *Not OK*.

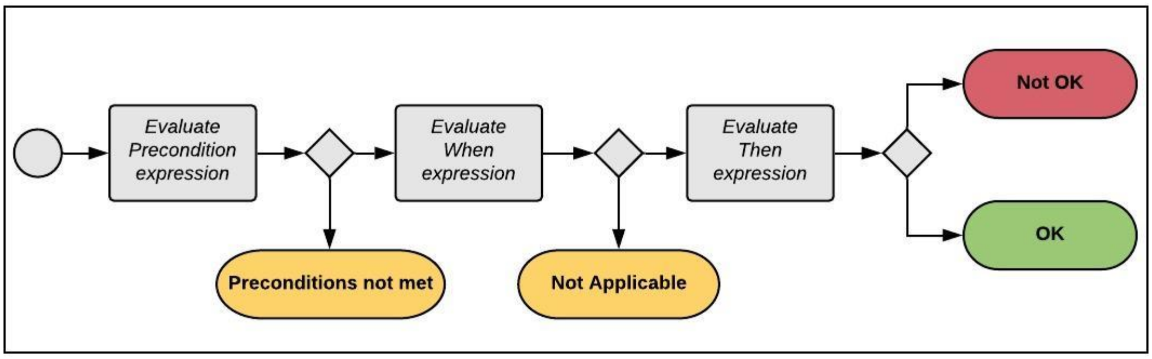


Figure 7.10: The expressions *preconditions*, *when* and *then* and the algorithm to detect the occurrence of an agile smell

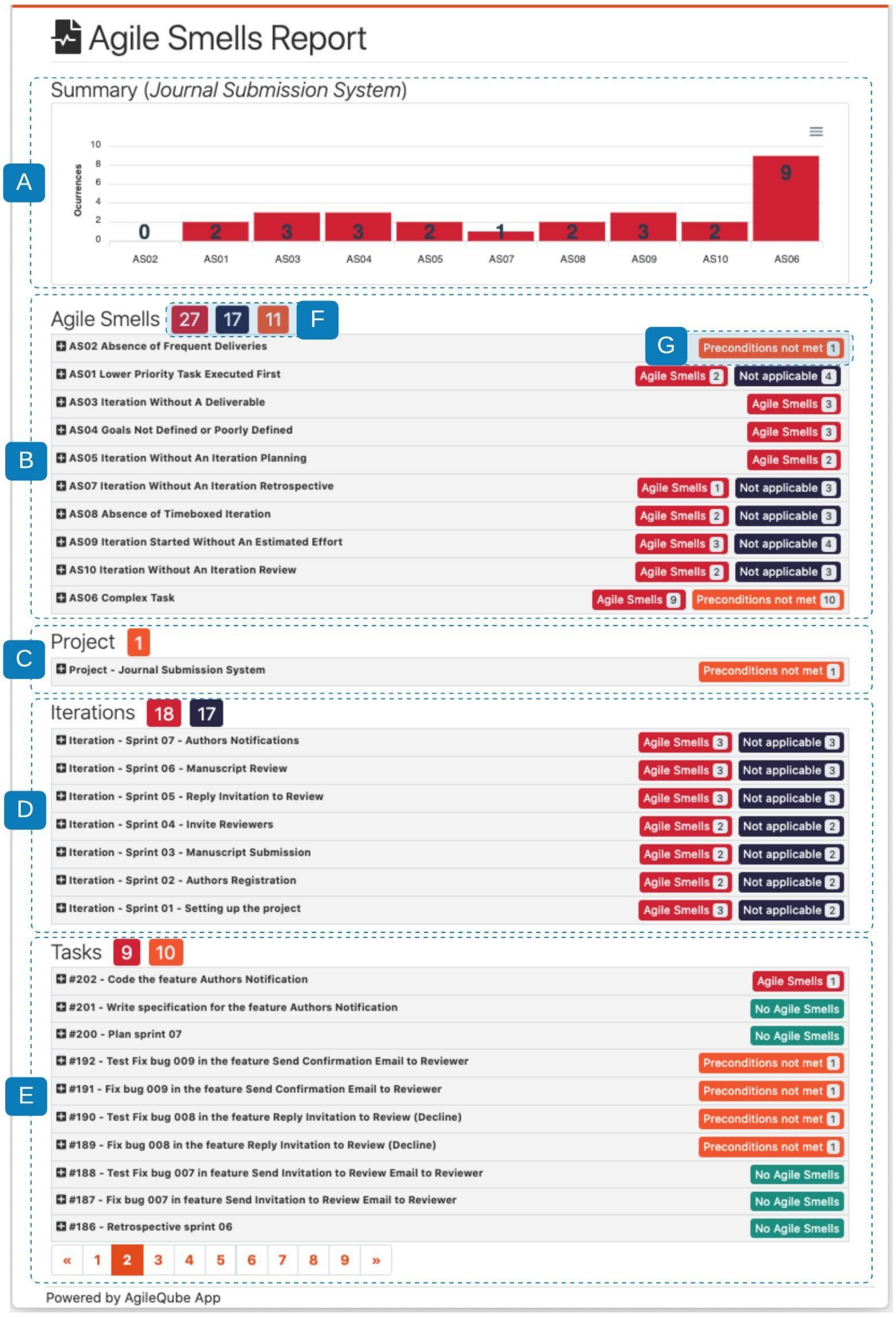
### *Agile Smells Report*

After executing the *Detection* phase, the *Detection Engine* produces an *Agile Smells Report* which is a report that indicates the agile smells that were detected in the *Project*, *Iteration* and *Task* elements. Figure 7.11 shows an overview of the report that is organized in 5 sections: (A) *Summary*; (B) *Agile Smells*; (C) *Project*; (D) *Iterations*; and (E) *Tasks*.

The section *Summary* presents a bar chart that indicates the distribution of occurrences per agile smell. Thus, it is possible to highlight the agile smells that have more occurrences in the project.

The section *Agile Smells* depicts the information presented in the bar chart and shows a table with the 10 agile smells and their corresponding occurrences in the project. The agile smells on this table are presented on the following order: first the agile smell related to the *Project* (*AS 02*), followed by the agile smells related to the *Iterations* (*AS 01* to *AS 10*) and finally the agile smell related to the *Tasks* (*AS 06*). The report also provides an expanded view for each agile smell that shows the target elements associated with the given agile smell (see Figure 7.12). The target elements are listed on the following order: first the elements where the agile smell was detected, followed by the elements were the agile smell was not applicable and finally the elements where the preconditions for detecting the agile smell were not met.

The sections *Project*, *Iterations* and *Tasks* show the corresponding target elements and the occurrences of agile smells associated with each of them. The report also provides an expanded view for each target element that lists the agile smells occurrences associated to the given target element (see Figure 7.13). The expanded view has three columns: (A) agile smell name; (B) result (*not ok*, *not applicable* or


Agile Smells Report

A

Summary (*Journal Submission System*)

Ocurrences

| AS02 | AS01 | AS03 | AS04 | AS05 | AS07 | AS08 | AS09 | AS10 | AS06 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 2 | 1 | 2 | 3 | 2 | 9 |

B

Agile Smells 27 17 11 F

G

| | |
|---|---|
| AS02 Absence of Frequent Deliveries | Preconditions not met 1 |
| AS01 Lower Priority Task Executed First | Agile Smells 2 Not applicable 4 |
| AS03 Iteration Without A Deliverable | Agile Smells 3 |
| AS04 Goals Not Defined or Poorly Defined | Agile Smells 3 |
| AS05 Iteration Without An Iteration Planning | Agile Smells 2 |
| AS07 Iteration Without An Iteration Retrospective | Agile Smells 1 Not applicable 3 |
| AS08 Absence of Timeboxed Iteration | Agile Smells 2 Not applicable 3 |
| AS09 Iteration Started Without An Estimated Effort | Agile Smells 3 Not applicable 4 |
| AS10 Iteration Without An Iteration Review | Agile Smells 2 Not applicable 3 |
| AS06 Complex Task | Agile Smells 9 Preconditions not met 10 |

C

Project 1

| | |
|---|---|
| Project - Journal Submission System | Preconditions not met 1 |

D

Iterations 18 17

| | |
|---|---|
| Iteration - Sprint 07 - Authors Notifications | Agile Smells 3 Not applicable 3 |
| Iteration - Sprint 06 - Manuscript Review | Agile Smells 3 Not applicable 3 |
| Iteration - Sprint 05 - Reply Invitation to Review | Agile Smells 3 Not applicable 3 |
| Iteration - Sprint 04 - Invite Reviewers | Agile Smells 2 Not applicable 2 |
| Iteration - Sprint 03 - Manuscript Submission | Agile Smells 2 Not applicable 2 |
| Iteration - Sprint 02 - Authors Registration | Agile Smells 2 Not applicable 2 |
| Iteration - Sprint 01 - Setting up the project | Agile Smells 3 Not applicable 2 |

E

Tasks 9 10

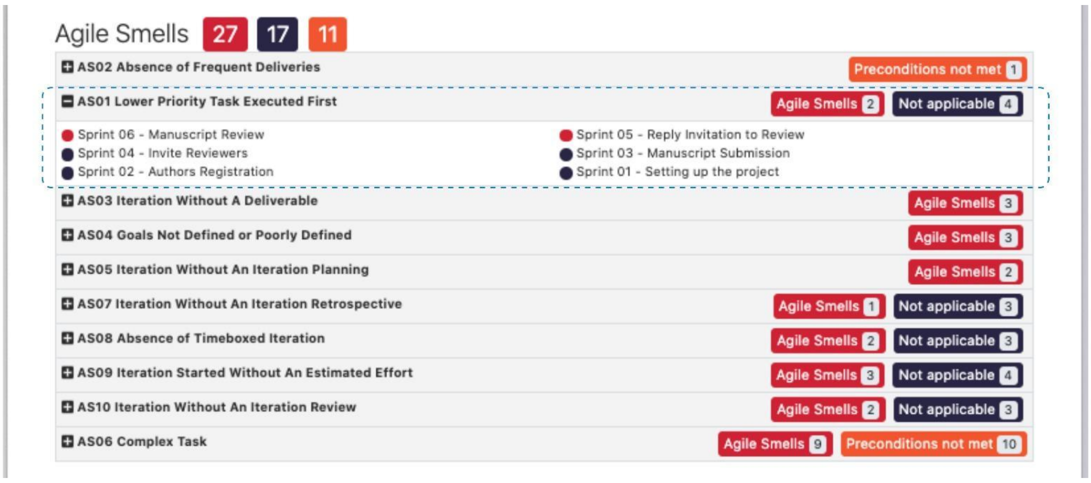
| | |
|---|---|
| #202 - Code the feature Authors Notification | Agile Smells 1 |
| #201 - Write specification for the feature Authors Notification | No Agile Smells |
| #200 - Plan sprint 07 | No Agile Smells |
| #192 - Test Fix bug 009 in the feature Send Confirmation Email to Reviewer | Preconditions not met 1 |
| #191 - Fix bug 009 in the feature Send Confirmation Email to Reviewer | Preconditions not met 1 |
| #190 - Test Fix bug 008 in the feature Reply Invitation to Review (Decline) | Preconditions not met 1 |
| #189 - Fix bug 008 in the feature Reply Invitation to Review (Decline) | Preconditions not met 1 |
| #188 - Test Fix bug 007 in feature Send Invitation to Review Email to Reviewer | No Agile Smells |
| #187 - Fix bug 007 in feature Send Invitation to Review Email to Reviewer | No Agile Smells |
| #186 - Retrospective sprint 06 | No Agile Smells |

« 1 2 3 4 5 6 7 8 9 »

Powered by AgileQube App

Figure 7.11: *Agile Smells Report* overview

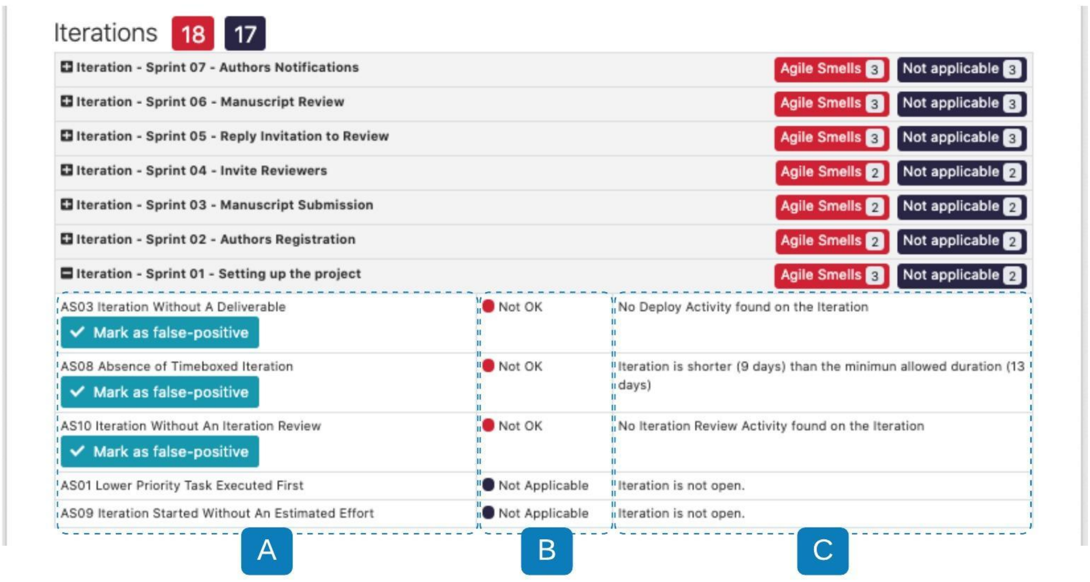
*preconditions not met*) and; (C) a description that gives a hint of the result. When the result is *not ok*, the first column also shows a button that allows marking that



Agile Smells 27 17 11

| | |
|---|---|
| AS02 Absence of Frequent Deliveries | Preconditions not met 1 |
| AS01 Lower Priority Task Executed First | Agile Smells 2 · Not applicable 4 |
| ● Sprint 06 - Manuscript Review<br>● Sprint 04 - Invite Reviewers<br>● Sprint 02 - Authors Registration | ● Sprint 05 - Reply Invitation to Review<br>● Sprint 03 - Manuscript Submission<br>● Sprint 01 - Setting up the project |
| AS03 Iteration Without A Deliverable | Agile Smells 3 |
| AS04 Goals Not Defined or Poorly Defined | Agile Smells 3 |
| AS05 Iteration Without An Iteration Planning | Agile Smells 2 |
| AS07 Iteration Without An Iteration Retrospective | Agile Smells 1 · Not applicable 3 |
| AS08 Absence of Timeboxed Iteration | Agile Smells 2 · Not applicable 3 |
| AS09 Iteration Started Without An Estimated Effort | Agile Smells 3 · Not applicable 4 |
| AS10 Iteration Without An Iteration Review | Agile Smells 2 · Not applicable 3 |
| AS06 Complex Task | Agile Smells 9 · Preconditions not met 10 |

Figure 7.12: *Agile Smells Report* - *Agile Smells* section



Iterations 18 17

| | |
|---|---|
| Iteration - Sprint 07 - Authors Notifications | Agile Smells 3 · Not applicable 3 |
| Iteration - Sprint 06 - Manuscript Review | Agile Smells 3 · Not applicable 3 |
| Iteration - Sprint 05 - Reply Invitation to Review | Agile Smells 3 · Not applicable 3 |
| Iteration - Sprint 04 - Invite Reviewers | Agile Smells 2 · Not applicable 2 |
| Iteration - Sprint 03 - Manuscript Submission | Agile Smells 2 · Not applicable 2 |
| Iteration - Sprint 02 - Authors Registration | Agile Smells 2 · Not applicable 2 |
| Iteration - Sprint 01 - Setting up the project | Agile Smells 3 · Not applicable 2 |

| A | B | C |
|---|---|---|
| AS03 Iteration Without A Deliverable<br>✓ Mark as false-positive | ● Not OK | No Deploy Activity found on the Iteration |
| AS08 Absence of Timeboxed Iteration<br>✓ Mark as false-positive | ● Not OK | Iteration is shorter (9 days) than the minimun allowed duration (13 days) |
| AS10 Iteration Without An Iteration Review<br>✓ Mark as false-positive | ● Not OK | No Iteration Review Activity found on the Iteration |
| AS01 Lower Priority Task Executed First | ● Not Applicable | Iteration is not open. |
| AS09 Iteration Started Without An Estimated Effort | ● Not Applicable | Iteration is not open. |

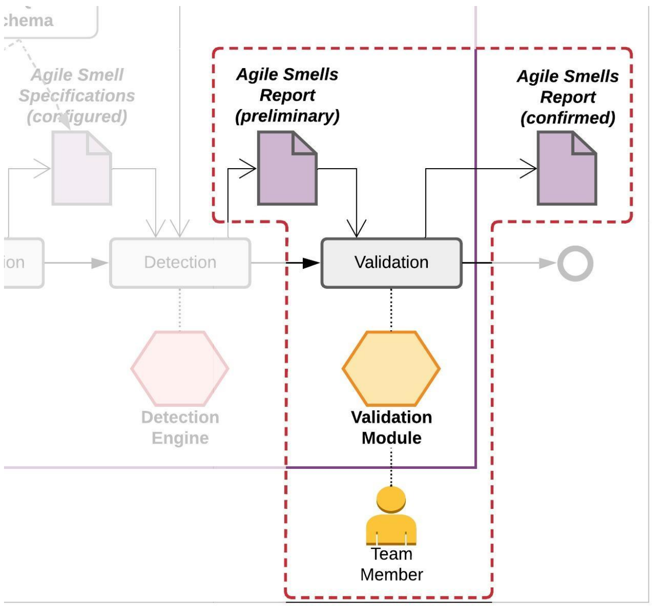
Figure 7.13: *Agile Smells Report* - *Iterations* section

occurrence as a false-positive (the feature is discussed in the next section).

The sections *Agile Smells*, *Project*, *Iterations* and *Tasks* also have labels (elements F and G) that summarize the number of occurrences of agile smells as follows: 1. *Agile Smells* (red label): indicates the number of agile smells detected; 2. *Not applicable* (black label): indicates the number of cases where the agile smells were not applicable; 3. *Preconditions not met* (orange label): the number of cases where the preconditions to detect the agile smell were not met; and 4. *No Agile Smells* (green label): the label is shown when the value of all previous labels are zero.

The *Agile Smells Report (preliminary)* is the input of the next and last phase, *Validation*.

### 7.3.4 *Validation Module*



:hema
Agile Smell Specifications (configured)
Agile Smells Report (preliminary)
Agile Smells Report (confirmed)
ion
Detection
Validation
Detection Engine
Validation Module
Team Member

Figure 7.14: *Validation Module* overview

The last component of the *AgileQube App* is the *Validation Module*, a module that is used in the *Validation* phase of the *AgileQube Approach*. As highlighted in Figure 7.14, this module enables a *Team Member* (or someone who knows the project context) to assess the *Agile Smells Report (preliminary)* and decide which of the detected agile smells are false-positive. A false-positive occurrence of an agile smell is a situation in which the agile smell is misdetected or the team deliberately accepts the situation that produces the agile smell. For example, a project may have some complex tasks that are difficult to break down into smaller tasks. In this case, the *Detection Engine* identifies the agile smell *Complex Tasks* in the tasks whose complexity exceeds the maximum limit defined in the specification. It is possible to use the *Validation Module* to individually assess each occurrence of the agile smell *Complex Tasks* and mark those unbreakable tasks as false-positive. The detection of false-positive agile smells may occur in other situations such as:

(a) Not timeboxed iterations that generate occurrences of the agile smell *Absence of Timeboxed Iteration* but that exceed the predefined duration for acceptable reasons not mapped in the project management system;

(b) Low priority tasks started at the beginning of an iteration that generate occurrences of the agile smell *Lower Priority Tasks Executed First* but are assigned to developers that do not have the skills required to perform high priority tasks.

Although the *Agile Smell Schema* offers an expressive mechanism to specify the agile smells and the use of parameters enables the configuration of context based variables, it is almost inevitable that the *Detection Engine* detects false-positive agile smells. The expertise and seniority of the team with agile development have a strong influence on this stage of the approach. A team with little or no experience with agile development tends to mark fewer items as false-positive. After all, applying agile practices in a straightforward manner is more suitable for beginning teams. On the other hand, a more experienced team is usually more flexible and creative in its development processes. Some situations detected as agile smells were deliberately introduced by the team. Thus, it is common for more experienced teams to mark more agile smells as false-positive. Figure 7.13 shows the interface that enables a *Team Member* to manually assess an occurrence of an agile smell and mark it as a false-positive.

## 7.4 Conclusion

This chapter presented the *AgileQube App*, a computational system that is part of the *AgileQube Approach* and that supports the phases *Specification*, *Configuration*, *Detection*, and *Validation.*

The *AgileQube App* is composed of four components: *(a) Specification Module*, *(b) ETL Module*, *(c) Detection Engine*, and *(d) Validation Module*. The *Specification Module* enables a *Programmer* to specify and configure agile smells using the

*Agile Smell Schema*. The *ETL Module* loads data directly from a project management platform, transforms the loaded data into an instance of the *Agile Project Metamodel* and sends the resulting data to the *Detection Engine*. The *Detection Engine* identifies agile smells in an agile project and generates an *Agile Smells Report (preliminary)*. The *Validation Module* enables a *Team Member* to assess the generated report and mark the false-positive occurrences of agile smells.

This contribution enables this research to address the goal G4 (*Define and implement a computational system*) by providing a computational system to support the *AgileQube Approach*.

# Chapter 8

# Case Studies

*This chapter focuses on demonstrating and validating the AgileQube Approach to automatically (or semi-automatically) detect agile smells in agile projects and is organized in the following sections: Introduction, Open-source projects hosted on GitHub, Open-source projects hosted on ZenHub, Journal Submission System, Terminal Operational System, and Conclusion.*

## 8.1 Introduction

In the previous chapters, we have presented the *AgileQube Approach* (Chapter 3) and its main components which include: the *Catalogue of Agile Smells* (Chapter 4), the *Agile Project Metamodel* (Chapter 5), the *Agile Smell Schema* (Chapter 6), and the *AgileQube App* (Chapter 7). We have also partially demonstrated and validated the research goals as follows:

1. In Section 4.5, we presented the *Catalogue of Agile Smells* that fulfills the goal G1 (*Define a catalogue of agile smells*);
2. In Section 6.5, we demonstrated how the *Agile Project Metamodel* and the *Agile Smell Schema* enable the specification of many different agile smells and, hence, validated the goal G2 (*Define an agile smell representation approach*).

This chapter presents the case studies that were conducted in the demonstration and validation phase of this research and that aim to validate whether the proposed approach is capable of automatically (or semi-automatically) detecting agile smells in agile projects.

## 8.2 Case Studies Methodology

The methodology of the case studies were based on the guidelines proposed by RUNESON and HÖST 2009 [52] and is organized in 5 major process phases:

*Phase 1 - Case study design*: the case is selected, objectives are defined and the case study is planned.

*Phase 2 - Preparation for data collection*: procedures and protocols for data collection are defined.

*Phase 3 - Data Collection*

*Phase 4 - Analysis of collected data*

*Phase 5 - Reporting*

## 8.3 Cases Selection

The selection of the agile projects that would be used in the case studies was a challenging endeavour in this research. We have tried four strategies:

1. Open-source projects hosted on *GitHub*;

2. Open-source projects hosted on *ZenHub*;

3. Simulated agile project hosted on *ZenHub*; and

4. Private and real agile project (migrated to *ZenHub*).

The remainder of this chapter presents and discusses the results of these strategies.

## 8.4 Open-source projects hosted on *GitHub*

Initially, our strategy was to conduct case studies using open-source (OS) projects hosted on *GitHub*. As the platform is the most popular repository for open-source projects (with more than 30 million active users and more than 100 million repositories (WARNER 2018 [207])), this strategy would allow us to reach a potential high number of projects.

We initiated this strategy by mapping the candidate projects, i.e., projects that met the following criteria: (C1) the project should be hosted on *GitHub*; (C2) the project should be public; (C3) the project should have issues (not only source-code);

and (C4) the issues should be organized in milestones. As detailed in Section 7.3.2, *GitHub issues* and *milestones* are mapped, respectively, to *tasks* and *iterations* in the *Agile Project Metamodel*. We used both the conventional Google search engine and the *GitHub* search engine to search for projects that met the mapping criteria. We have found and analyzed many projects but this strategy showed unsuccessful for the following reasons:

1. The number of projects that met criteria C1 and C2 was high but few of these projects met criteria C3 and C4;

2. The *issues* and *milestones* of the candidate projects did not have the information required to perform the detection of agile smells. For example, the issues on these projects usually have only a description and a few labels but no information related to: (a) their priority, (b) effort estimation, (c) start and finish dates, (d) workflow status, and (e) activity in the process. In this scenario, the *Agile Smells Report (preliminary)* generated by the *Detection Engine* contained only *preconditions not met* occurrences.

These limitations revealed that we needed a more effective way to find projects hosted on *GitHub* that use the platform not only as source-code repository but also as project management system. As this attempt stopped in phase 1 (*Case study design*), we decided to try a second strategy: selecting open-source software projects hosted on *ZenHub* to validate the proposed approach.

## 8.5 Open-source projects hosted on *ZenHub*

The second strategy consisted in selecting open-source software projects hosted on *ZenHub* to validate the proposed approach. As the platform is focused on project management (*ZenHub* is one of the most popular *GitHub* extensions for project management (BUTLER and PAQUETTE 2016 [211])), we estimated that the issues of the selected projects had the information required to perform the detection of agile smells. To select the projects, we adapted the mapping criteria mentioned above as follows: (C1) the project should be hosted on *ZenHub*; (C2) the project should be public; (C3) the project should have issues; and (C4) the issues should be organized in milestones. A challenge faced at the mapping projects phase is that *ZenHub* does not provide a search engine to find projects hosted on the platform. Thus, although the platform has more than 3000 public projects, it was only possible to access them if you were a project member or if you had the *ZenHub* project access code. To overcome this limitation, we contacted the *ZenHub* support team and after explaining the purpose of our research, they provided us access to two public projects:

1. **Mozilla/Bedrock** [1] (issues: approx. 2,000; milestones: approx. 20)
2. **Google/Flutter** [2] (issues: approx. 40,000; milestones: approx. 60)

The *Mozilla/Bedrock* project aims to maintain the *mozilla.org* website (*Bedrock* is the code name of *mozilla.org*). The project had around 2,000 issues and 20 milestones when we performed the agility assessment. The *ETL Module* was able to load the data from *ZenHub* and transform the extracted data to the *Agile Project Metamodel*. A limitation observed is that the *ZenHub* API, after receiving a certain number of requests from the same host in a short time period, blocked new requests from that host for 60 seconds. During this time, all requests from the blocked host receive an error response *403 Forbidden/API Rate limit reached.* To overcome this limitation, we programmed the *ETL Module* to "wait" for 60 seconds when this situation happens. As the project had a high number of issues, the overall time to load the data from *ZenHub* ranged between 30 to 40 minutes. Although this research has not established any response time requirements, the data load time of the *Mozilla/Bedrock* project was a threat to validate the *AgileQube Approach*. In addition, similar to what was observed in *GitHub* projects, most of the project's issues did not contain the information required to detect agile smells.

The *Google/Flutter* project aims to maintain the framework of the same name owned by Google to create mobile, web, and desktop applications from a single codebase. The project had about 40,000 issues distributed over 60 milestones when we performed the agility assessment. The *ETL Module* was able to load the data from *ZenHub* and transform the extracted data to the *Agile Project Metamodel* but, due to the API rate limitations above discussed, the overall time to load the data ranged between 150 to 180 minutes. The data load time of the *Google/Flutter* project made almost impracticable to use this project to validate the *AgileQube Approach*.

The attempts of using the OS projects presented above have revealed unsuccessful mostly because the issues of these projects did not have the information

[1] `https://app.zenhub.com/workspaces/websites-team-59bd527821e82e515786ca73/board?repos=1616665`

[2] `https://app.zenhub.com/workspaces/flutter-add-to-app-58ed3fad669ec5806a2cbbae/board?repos=31792824`

required to detect agile smells such as: 1. priority, 2. effort estimation, 3. start and finish dates, 4. workflow status, and 5. activity in the process. The lack of this information can be explained by the fact that OS projects, unlike conventional agile projects, generally do not have a project plan, schedule or delivery list (MOCKUS *et al.* 2000 [212]).

As we could not explore the use of the proposed approach to detect agile smells in OS projects, we decided to apply other two strategies to validate the *AgileQube Approach*: (a) use a simulated agile project, *Journal Submission System*, (Section 8.6) and (b) use a private and real agile project, *Terminal Operational System*, (Section 8.7).

## 8.6 *Journal Submission System*

The third strategy in our endeavor to validate the *AgileQube Approach* was to conduct a case study with a simulated agile project. Given the challenges faced in our attempts to use OS projects to demonstrate the proposed approach, such a strategy was fundamental to enable the first experiment involving the use of the *AgileQube Approach* to detect agile smells.

### 8.6.1 Phase 1: Case study design

#### Case Study Objective, Research Question, and validation criteria

The purpose of the case study is to answer the following research question:

***RQ:*** *How suitable is the AgileQube Approach to be used with the purpose of identifying agile smells in the selected agile project?*

To answer RQ, we observed two criteria:

1. The recall and precision of the *AgileQube Approach*;
2. The computational time to generate an *Agile Smells Report*.

We borrowed the precision and recall measures from the field of information retrieval (GROSSMAN and FRIEDER 2012 [213]) and from studies aimed at evaluating and comparing code smell detection tools (MOHA *et al.* 2009 [139], FERNANDES *et al.* 2016 [214], PAIVA *et al.* 2017 [215]) and adapted them in the following way:

(a) precision denotes the number of true agile smells identified among the detected agile smells, while

(b) recall denotes the number of detected agile smells among the existing agile smells.

$$precision = \frac{|\{existing\ agile\ smells\} \cap \{detected\ agile\ smells\}|}{|\{detected\ agile\ smells\}|}$$

$$recall = \frac{|\{existing\ agile\ smells\} \cap \{detected\ agile\ smells\}|}{|\{existing\ agile\ smells\}|}$$

Computational time refers only to the time spent in loading data from the project management system and generating the *Agile Smells Report* (*Detection* phase). Thus, the time spent in specifying the agile smells and configuring the project in the *AgileQube App* (*Specification* and *Configuration* phases) is not considered.

Issues related to the usability of the approach or the applicability of the results are not subject of this validation. These aspects, although relevant, will not be considered in this research.

**About the *Journal Submission System* project**

We selected an agile project, that we named *Journal Submission System*, to validate the detection of agile smells by the proposed approach according to the validation protocol presented previously. The project aims the development of a web-based Manuscrit Submission Management System (MSMS) that simplifies and facilitates collecting, tracking and management of academic research paper submissions. Note that the *Journal Submission System* does not represent a real project. Hence, there is no development team or source-code.

The design of the *Journal Submission System* is based on the model proposed in (JACKSI 2015 [216]) that has 6 main features and 3 key users:

**Main features**:

1. *Authors Registration*;
2. *Manuscript Submission*;
3. *Invite Reviewers*;
4. *Reply Invitation to Review*;

5. *Review Manuscript*; and
6. *Notify Editor and Authors*

**Key users**:

1. *Authors*,
2. *Editors*, and
3. *Reviewers*

A detailed description of these features is not relevant to the analysis conducted in this case study. For this reason, we have not included a full description of these features and key users in this text.

## Project Structure

Project structure, in the context of this experiment, denotes the iterations and the tasks that compose the project. Since the *Journal Submission System* is a simulated project, we had to create its project structure. To mitigate the bias of this strategy, two independent Project Managers (PM) aided the definition of this structure. These professionals have a large experience with agile development (more than 10 years) and both are PMP certified[3]. One of them has a Master degree.

To define the project structure, we followed, along with the PMs, the steps below:

(1) We analyzed the project main features and broke them into more detailed features;

(2) We defined the tasks necessary to implement each detailed feature;

(3) We included generic tasks such as those related to project setup, iteration planning, iteration review and release deployment;

(4) We included tasks related to bugs and change requests; and

(5) We defined the iterations and distributed the tasks over them (AKA Iteration Planning).

It is not usual to plan multiple iterations ahead in an agile project. However, as this is a case study with a simulated project, we adopted this strategy to build the simulation project structure required for data collection. These steps were performed collaboratively through a Google Drive document. The resulting project structure had a total of 109 tasks distributed over the following 7 iterations:

1. *Sprint 1 - Setting up the project*: 10 tasks/10 days;

[3]`https://www.pmi.org/certifications/project-management-pmp`

2. *Sprint 2 - Authors registration*: 10 tasks/10 days;
3. *Sprint 3 - Manuscript submission*: 15 issues/20 days;
4. *Sprint 4 - Invite reviewers*: 21 issues/15 days;
5. *Sprint 5 - Reply invitation to review*: 21 issues/13 days;
6. *Sprint 6 - Manuscript review*: 21 issues/17 days;
7. *Sprint 7 - Authors notification*: 12 issues/15 days.

Next, we created public projects on *GitHub*[4] and *ZenHub*[5] and entered the project structure according to the approach proposed in Section 7.3.2. The 7 milestones and the 109 issues that compose the project are shown in Table F.1 of Appendix F. The issue identifiers range from #50 to #212 with gaps between some identifiers and, therefore, the greatest identifier does not represent the number of issues.

### Agile Smells Injection

To simulate the agile smells, we applied a technique based on bug injection, which consists of deliberately injecting bugs into the source-code for evaluating the effectiveness of bug finders. Bug injection as a validating approach has been extensively explored in the domain of traditional programs (PEWNY and HOLZ 2016 [217], DOLAN-GAVITT *et al.* 2016 [218], BONETT *et al.*, GHALEB and PATTABIRAMAN 2018, 2020 [219, 220]).

We intentionally injected the following agile smells into the *Journal Submission System* project:

(a) There are low priority tasks executed before high priority tasks in the iterations *Sprint 05* and *Sprint 06* (*Lower Priority Tasks Executed First*);

(b) There is no deliverable in the iterations *Sprint 01*, *Sprint 04* and *Sprint 07* (*Iteration Without a Deliverable*);

(c) There is no goal defined for the iteration *Sprint 02* and the goals defined for the iterations *Sprint 05* and *Sprint 07* are too short (less than the required length defined on the agile smell specification) (*Goals Not Defined or Poorly Defined*);

[4]https://github.com/utelemaco/journal-submission-system/issues
[5]https://app.zenhub.com/workspaces/project-a-5f40b4508f8d67000f5fde0b

Table 8.1: Agile smells injected in the JSS project

| **Agile Smell** | **Target** | **Occurrences** |
|---|---|---|
| (AS 01) *Lower Priority Tasks Executed First* | Sprint 05<br>Sprint 06 | 2 |
| (AS 02) *Absence of Frequent Deliveries* | | 0 |
| (AS 03) *Iteration Without a Deliverable* | Sprint 01<br>Sprint 04<br>Sprint 07 | 3 |
| (AS 04) *Goals Not Defined or Poorly Defined* | Sprint 02<br>Sprint 05<br>Sprint 07 | 3 |
| (AS 05) *Iteration Without an Iteration Planning* | Sprint 02<br>Sprint 06 | 2 |
| (AS 06) *Complex Tasks* | #51 #52 #53<br>#72 #92 #119<br>#148 #172 #202 | 9 |
| (AS 07) *Iteration Without an Iteration Retrospective* | Sprint 03 | 1 |
| (AS 08) *Absence of Timeboxed Iteration* | Sprint 01<br>Sprint 03 | 2 |
| (AS 09) *Iteration Started without an Estimated Effort* | Sprint 05<br>Sprint 06<br>Sprint 07 | 3 |
| (AS 10) *Iteration Without an Iteration Review* | Sprint 01<br>Sprint 04 | 2 |
| | **Total** | **27** |

(d) There is no planning in the iterations *Sprint 02* and *Sprint 06* (*Iteration Without an Iteration Planning*);

(e) There is no retrospective in the iteration *Sprint 03* (*Iteration Without an Iteration Retrospective*);

(f) The iterations *Sprint 01* and *Sprint 03* are not timeboxed (*Absence of Timeboxed Iteration*);

(g) The iterations *Sprint 05*, *Sprint 06* and *Sprint 07* started with no estimated tasks (*Iteration Started without an Estimated Effort*);

(h) There is no review in the iterations *Sprint 01* and *Sprint 04* (*Iteration Without an Iteration Review*);

(i) There are complex tasks in the project: *#51*, *#52*, *#53*, *#72*, *#92*, *#119*, *#148*, *#172* and *#202* (*Complex Tasks*);

A summary of the agile smells injected in the *Journal Submission System* project are presented in Table 8.1.

### 8.6.2 Phase 2: Preparation for data collection

The preparation for data collection in this case study consists of indicating to the *ETL Module* how to load the project data from *ZenHub*. The project configuration, that is shown in Figure 8.1, contains the following information:

(A) Github and Zenhub identifiers and API access tokens;

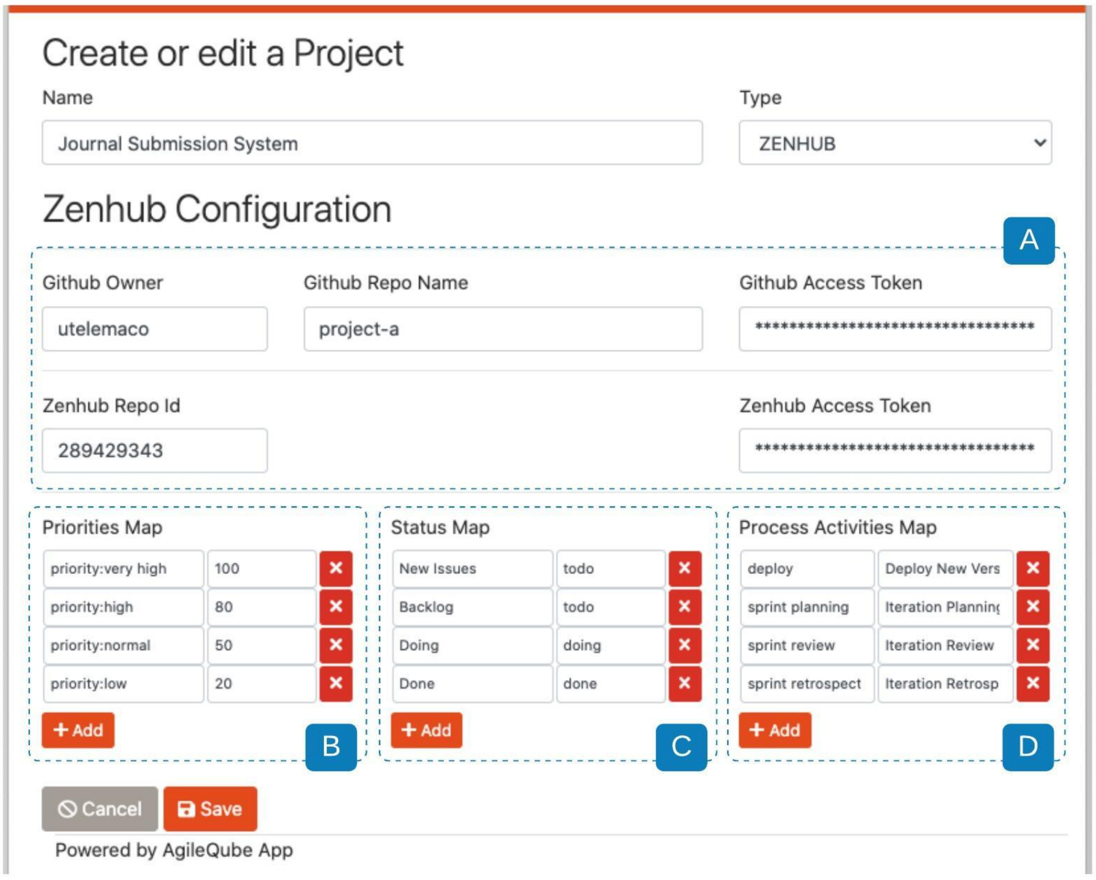
(B) Priorities Map;

(C) Status Map; and

(D) Process Activities Map.

There are four labels to represent task priorities (*priority:very high*, *priority:high*, *priority:normal*, and *priority:low*), four status (*New Issues* and *Backlog* that map to *todo*, *Doing*, and *Done*) and four labels to represent the activities from the software process (*deploy* that corresponds to the activity *Deploy New version*, *sprint planning* that corresponds to the activity *Iteration Planning*, *sprint review* that corresponds to *Iteration Review*, and *sprint retrospective* that corresponds to *Iteration Retrospective*).

This configuration enables the *ETL Module* to load project data from *ZenHub*, i.e. to access the platform, extract the data, transform the extracted data and send the transformed data to the detection engine as described in Section 7.3.2.


Create or edit a Project

Name: Journal Submission System

Type: ZENHUB

Zenhub Configuration

A

Github Owner: utelemaco

Github Repo Name: project-a

Github Access Token: **********************************

Zenhub Repo Id: 289429343

Zenhub Access Token: **********************************

Priorities Map (B)

| | |
|---|---|
| priority:very high | 100 |
| priority:high | 80 |
| priority:normal | 50 |
| priority:low | 20 |

Status Map (C)

| | |
|---|---|
| New Issues | todo |
| Backlog | todo |
| Doing | doing |
| Done | done |

Process Activities Map (D)

| | |
|---|---|
| deploy | Deploy New Vers |
| sprint planning | Iteration Planning |
| sprint review | Iteration Review |
| sprint retrospect | Iteration Retrosp |

Figure 8.1: JSS project configuration in the *AgileQube App*

### 8.6.3 Phase 3: Data Collection

After configuring the JSS project, the *AgileQube App* was able to load the project data from *ZenHub*, detect the agile smells and generate an *Agile Smells Report*.

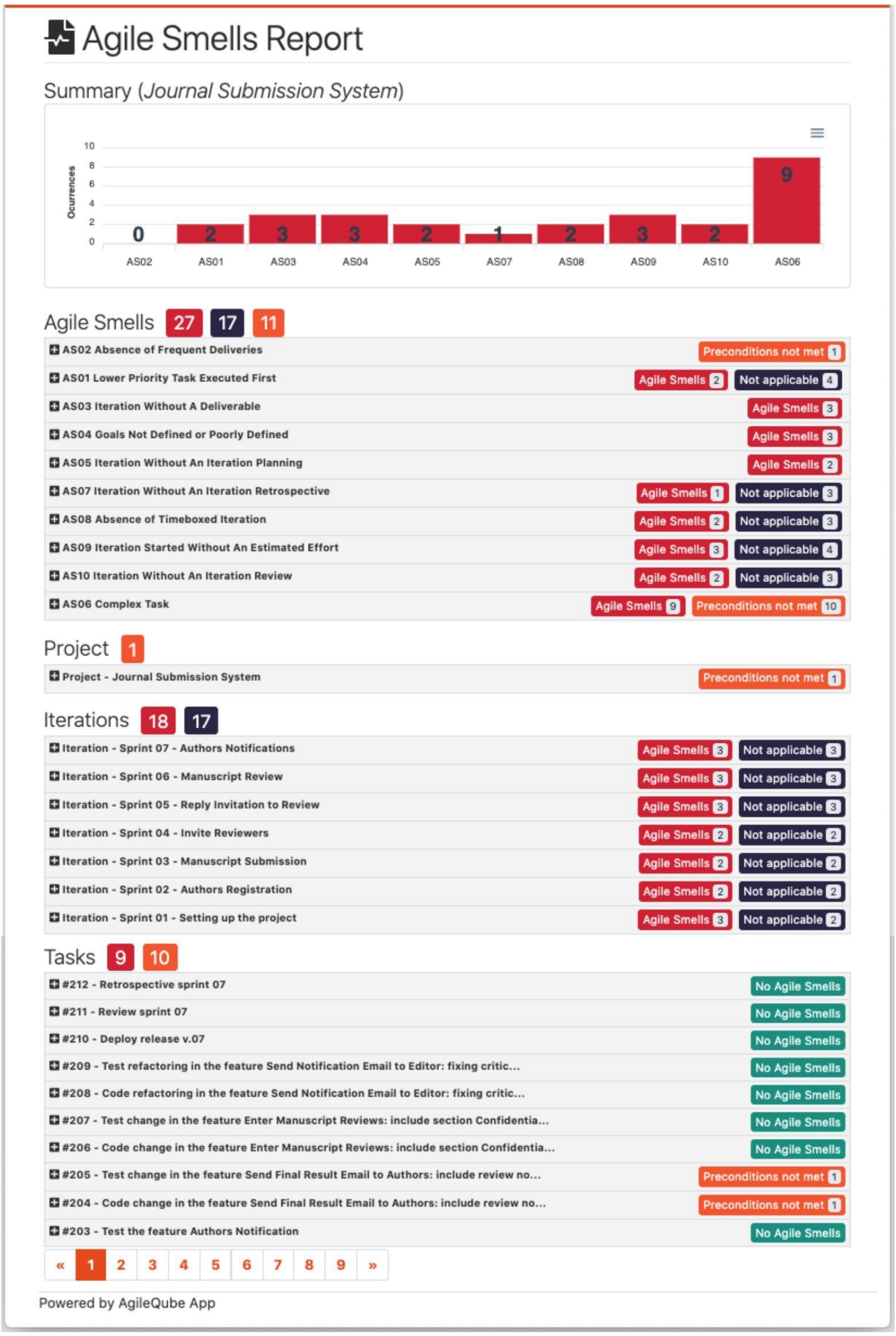
Figure 8.2 shows an overview of the generated report that can be divided in 5 sections: *Summary*, *Agile Smells*, *Project*, *Iterations*, and *Tasks*.

### 8.6.4 Phase 4: Analysis of collected data

The bar chart in the section *Summary* presents the distribution of agile smells occurrences over the project and reveals that: *(a)* the agile smell *AS 06 Complex Tasks* has the highest number of occurrences (9 in total); *(b)* the agile smells *AS 03 Iteration Without a Deliverable*, *AS 04 Goals Not Defined or Poorly Defined* and *AS 09 Iteration Started without an Estimated Effort* have 3 occurrences each; *(c)* the agile smells *AS 01 Lower Priority Tasks Executed First*, *AS 05 Iteration Without an Iteration Planning*, *AS 08 Absence of Timeboxed Iteration* and *AS 10 Iteration Without an Iteration Review* have 2 occurrences each; *(d)* the agile smell *AS 07 Iteration Without an Iteration Retrospective* has 1 occurrence; and *(e)* the agile smell *AS 02 Absence of Frequent Deliveries* has no occurrences.

Section *Agile Smells* presents the 10 agile smells and their corresponding occurrences. For example, the agile smell *AS 01 Lower Priority Tasks Executed First* has 2 occurrences and 4 not applicable cases. The agile smell *AS 05 Iteration Without an Iteration Planning* has 2 occurrences. The agile smell *AS 07 Iteration Without an Iteration Retrospective* has 1 occurrence and 3 not applicable cases. The agile smell *AS 02 Absence of Frequent Deliveries* has 1 case where the preconditions to detect the agile smell were not met. Section *Agile Smells* also shows in its title a summary of the total numbers of detected agile smells (27 agile smells occurrences, 17 not applicable cases, 11 cases where the preconditions for detection were not met).

An expanded view of section *Agile Smells* is shown in Figure 8.3. In the expanded view, it is possible to check the target elements associated with each agile smell. For example, the agile smell *AS 01 Lower Priority Tasks Executed First* is detected in the iterations *Sprint 06* and *Sprint 05* but is considered not applicable for the iterations *Sprint 04*, *Sprint 03*, *Sprint 02* and *Sprint 01*. The agile smell *AS 08 Absence of Timeboxed Iteration* is detected in the iterations *Sprint 03* and *Sprint 01* but is considered not applicable for the iterations *Sprint 07*, *Sprint 06*, and *Sprint 05*.



Agile Smells Report

Summary (*Journal Submission System*)

Agile Smells 27 17 11

| | |
|---|---|
| AS02 Absence of Frequent Deliveries | Preconditions not met 1 |
| AS01 Lower Priority Task Executed First | Agile Smells 2 Not applicable 4 |
| AS03 Iteration Without A Deliverable | Agile Smells 3 |
| AS04 Goals Not Defined or Poorly Defined | Agile Smells 3 |
| AS05 Iteration Without An Iteration Planning | Agile Smells 2 |
| AS07 Iteration Without An Iteration Retrospective | Agile Smells 1 Not applicable 3 |
| AS08 Absence of Timeboxed Iteration | Agile Smells 2 Not applicable 3 |
| AS09 Iteration Started Without An Estimated Effort | Agile Smells 3 Not applicable 4 |
| AS10 Iteration Without An Iteration Review | Agile Smells 2 Not applicable 3 |
| AS06 Complex Task | Agile Smells 9 Preconditions not met 10 |

Project 1

| | |
|---|---|
| Project - Journal Submission System | Preconditions not met 1 |

Iterations 18 17

| | |
|---|---|
| Iteration - Sprint 07 - Authors Notifications | Agile Smells 3 Not applicable 3 |
| Iteration - Sprint 06 - Manuscript Review | Agile Smells 3 Not applicable 3 |
| Iteration - Sprint 05 - Reply Invitation to Review | Agile Smells 3 Not applicable 3 |
| Iteration - Sprint 04 - Invite Reviewers | Agile Smells 2 Not applicable 2 |
| Iteration - Sprint 03 - Manuscript Submission | Agile Smells 2 Not applicable 2 |
| Iteration - Sprint 02 - Authors Registration | Agile Smells 2 Not applicable 2 |
| Iteration - Sprint 01 - Setting up the project | Agile Smells 3 Not applicable 2 |

Tasks 9 10

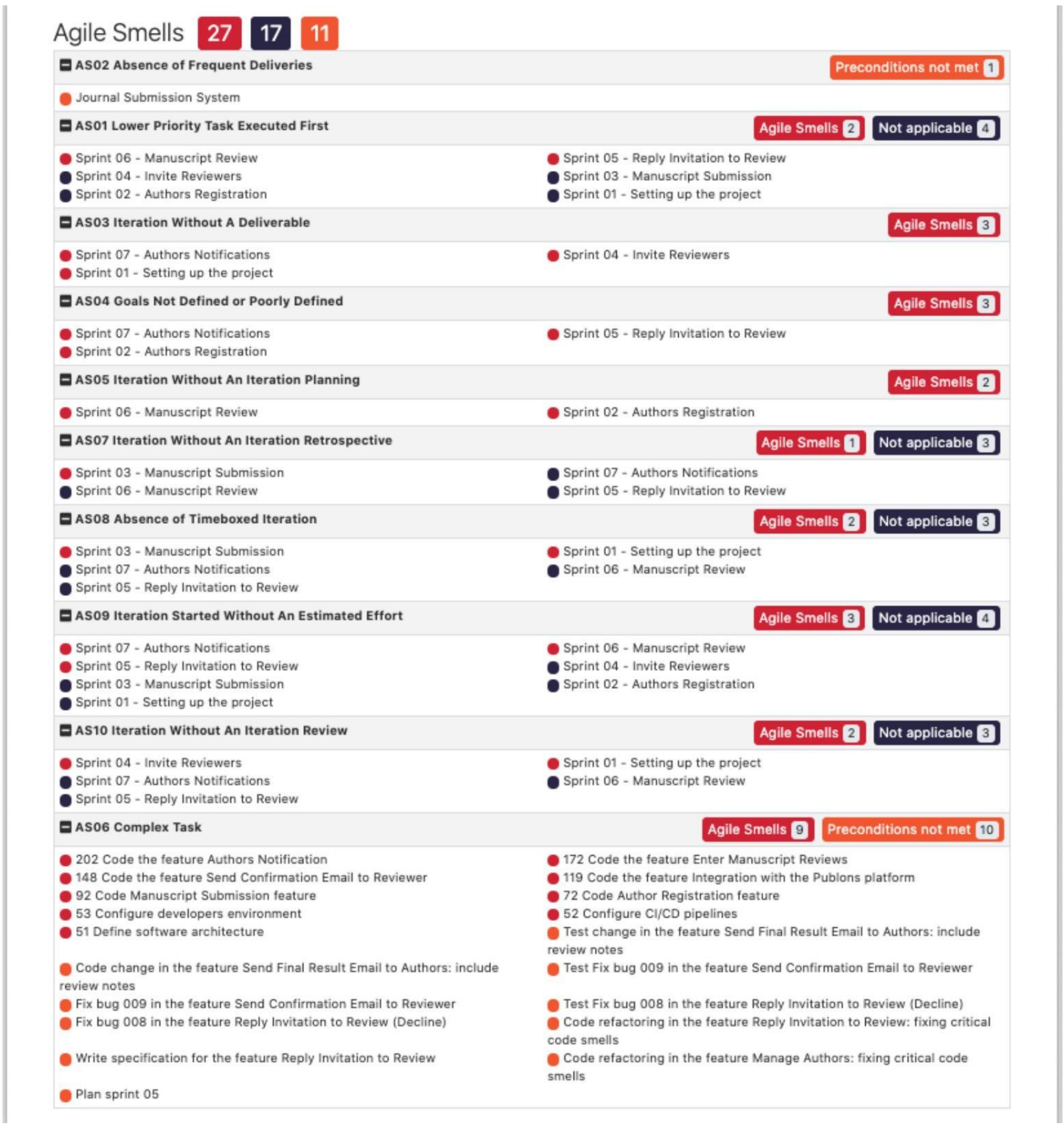
| | |
|---|---|
| #212 - Retrospective sprint 07 | No Agile Smells |
| #211 - Review sprint 07 | No Agile Smells |
| #210 - Deploy release v.07 | No Agile Smells |
| #209 - Test refactoring in the feature Send Notification Email to Editor: fixing critic... | No Agile Smells |
| #208 - Code refactoring in the feature Send Notification Email to Editor: fixing critic... | No Agile Smells |
| #207 - Test change in the feature Enter Manuscript Reviews: include section Confidentia... | No Agile Smells |
| #206 - Code change in the feature Enter Manuscript Reviews: include section Confidentia... | No Agile Smells |
| #205 - Test change in the feature Send Final Result Email to Authors: include review no... | Preconditions not met 1 |
| #204 - Code change in the feature Send Final Result Email to Authors: include review no... | Preconditions not met 1 |
| #203 - Test the feature Authors Notification | No Agile Smells |

« 1 2 3 4 5 6 7 8 9 »

Powered by AgileQube App

Figure 8.2: *Agile Smells Report* overview of the JSS project

Section *Project* presents the agile smells whose target element is the *Project*. Only the agile smell *AS 02 Absence of Frequent Deliveries* appears in this category.


Agile Smells 27 17 11

**AS02 Absence of Frequent Deliveries** — Preconditions not met 1

- 🟠 Journal Submission System

**AS01 Lower Priority Task Executed First** — Agile Smells 2 Not applicable 4

- 🔴 Sprint 06 - Manuscript Review
- ⚫ Sprint 04 - Invite Reviewers
- ⚫ Sprint 02 - Authors Registration
- 🔴 Sprint 05 - Reply Invitation to Review
- ⚫ Sprint 03 - Manuscript Submission
- ⚫ Sprint 01 - Setting up the project

**AS03 Iteration Without A Deliverable** — Agile Smells 3

- 🔴 Sprint 07 - Authors Notifications
- 🔴 Sprint 01 - Setting up the project
- 🔴 Sprint 04 - Invite Reviewers

**AS04 Goals Not Defined or Poorly Defined** — Agile Smells 3

- 🔴 Sprint 07 - Authors Notifications
- 🔴 Sprint 02 - Authors Registration
- 🔴 Sprint 05 - Reply Invitation to Review

**AS05 Iteration Without An Iteration Planning** — Agile Smells 2

- 🔴 Sprint 06 - Manuscript Review
- 🔴 Sprint 02 - Authors Registration

**AS07 Iteration Without An Iteration Retrospective** — Agile Smells 1 Not applicable 3

- 🔴 Sprint 03 - Manuscript Submission
- ⚫ Sprint 06 - Manuscript Review
- ⚫ Sprint 07 - Authors Notifications
- ⚫ Sprint 05 - Reply Invitation to Review

**AS08 Absence of Timeboxed Iteration** — Agile Smells 2 Not applicable 3

- 🔴 Sprint 03 - Manuscript Submission
- ⚫ Sprint 07 - Authors Notifications
- ⚫ Sprint 05 - Reply Invitation to Review
- 🔴 Sprint 01 - Setting up the project
- ⚫ Sprint 06 - Manuscript Review

**AS09 Iteration Started Without An Estimated Effort** — Agile Smells 3 Not applicable 4

- 🔴 Sprint 07 - Authors Notifications
- 🔴 Sprint 05 - Reply Invitation to Review
- ⚫ Sprint 03 - Manuscript Submission
- ⚫ Sprint 01 - Setting up the project
- 🔴 Sprint 06 - Manuscript Review
- ⚫ Sprint 04 - Invite Reviewers
- ⚫ Sprint 02 - Authors Registration

**AS10 Iteration Without An Iteration Review** — Agile Smells 2 Not applicable 3

- ⚫ Sprint 04 - Invite Reviewers
- ⚫ Sprint 07 - Authors Notifications
- ⚫ Sprint 05 - Reply Invitation to Review
- 🔴 Sprint 01 - Setting up the project
- ⚫ Sprint 06 - Manuscript Review

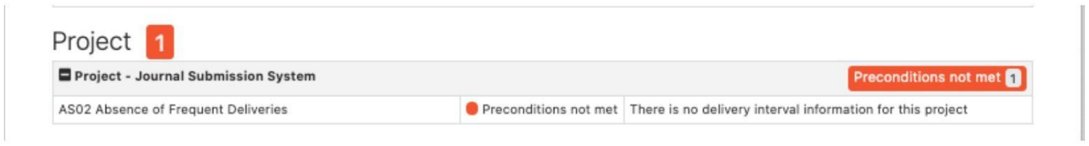
**AS06 Complex Task** — Agile Smells 9 Preconditions not met 10

- 🔴 202 Code the feature Authors Notification
- 🔴 148 Code the feature Send Confirmation Email to Reviewer
- 🔴 92 Code Manuscript Submission feature
- 🔴 53 Configure developers environment
- 🔴 51 Define software architecture
- 🟠 Code change in the feature Send Final Result Email to Authors: include review notes
- 🟠 Fix bug 009 in the feature Send Confirmation Email to Reviewer
- 🟠 Fix bug 008 in the feature Reply Invitation to Review (Decline)
- 🟠 Write specification for the feature Reply Invitation to Review
- 🟠 Plan sprint 05
- 🔴 172 Code the feature Enter Manuscript Reviews
- 🔴 119 Code the feature Integration with the Publons platform
- 🔴 72 Code Author Registration feature
- 🔴 52 Configure CI/CD pipelines
- 🟠 Test change in the feature Send Final Result Email to Authors: include review notes
- 🟠 Test Fix bug 009 in the feature Send Confirmation Email to Reviewer
- 🟠 Test Fix bug 008 in the feature Reply Invitation to Review (Decline)
- 🟠 Code refactoring in the feature Reply Invitation to Review: fixing critical code smells
- 🟠 Code refactoring in the feature Manage Authors: fixing critical code smells

Figure 8.3: *Agile Smells Report* of the JSS project - *Agile Smells* section

An expanded view of the section *Project* (see Figure 8.4) reveals the preconditions to detect the agile smell *AS 02* were not met since there is no delivery interval information for the given project.

Section *Iterations* presents the 7 iterations and their corresponding agile smells. For each iteration, it is shown labels indicating: the number of agile smells (*Agile Smells*), the number of not applicable cases (*Not Applicable*), the number of cases where the preconditions to detect the agile smell were not met (*Preconditions not met*) or a label indicating all previous numbers are zero (*No Agile Smells*). For example, iteration *Sprint 06* has 3 agile smells and 3 not applicable cases. Iteration *Sprint 01* has 3 agile smells and 2 not applicable cases. Section *Iterations* also shows a summary with the total numbers of detected agile smells in the iterations (18 agile



Project 1

| Project - Journal Submission System | | Preconditions not met 1 |
|---|---|---|
| AS02 Absence of Frequent Deliveries | ● Preconditions not met | There is no delivery interval information for this project |

Figure 8.4: *Agile Smells Report* of the JSS project - *Project* section

smells occurrences and 17 not applicable cases).

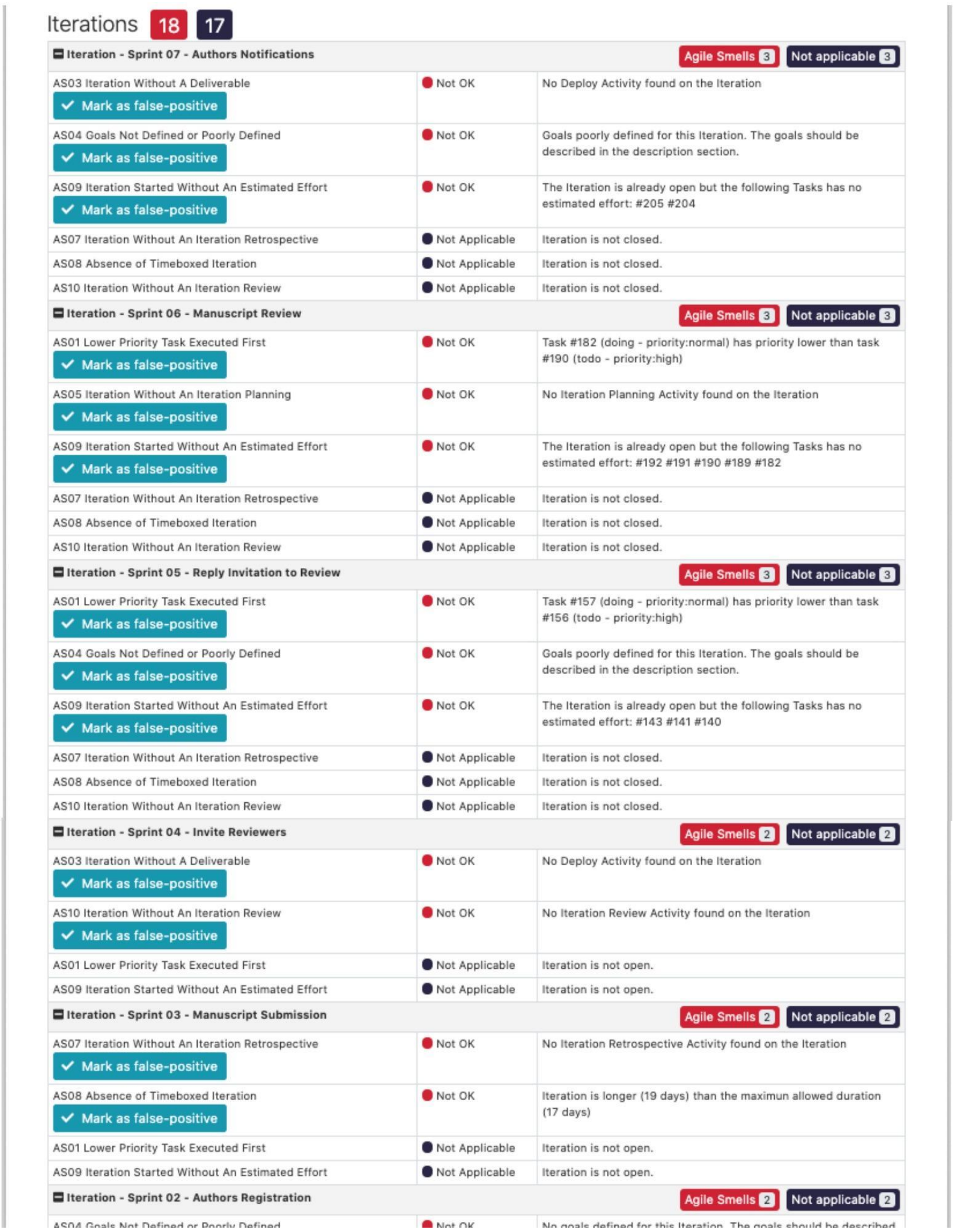
An expanded view of section *Iterations* is shown in Figure 8.5. In the expanded view, it is possible to see details of the agile smell occurrence such as: agile smell name, result and description. For example, the three agile smells detected in iteration *Sprint 06* are *AS 01*, *AS 05*, and *AS 09*. *AS 01* was detected because a normal priority task (*#182*) is being executed before a high priority task (*#190*). *AS 04* was detected because no planning activity was found in the iteration. *AS 09* was detected because the iteration is open but the tasks *#182*, *#189*, *#190*, *#191*, and *#192* were not estimated. The agile smells *AS 07*, *AS 08*, and *AS 10* are not applicable for *Sprint 06* because the iteration is not closed.

Section *Tasks* lists the 109 tasks and their corresponding agile smells. Similarly to section *Iterations*, this section shows, for each task, labels indicating the number of agile smells associated with the task (*Agile Smells*, *Not applicable*, *Preconditions not met*, *No Agile Smell*). A pagination component is enabled when the list of tasks reaches a specific number of elements. According to the section *Tasks* shown in Figure 8.2, 2 not applicable cases were detected in tasks *#205* and *#204*. No agile smells were detected in the tasks *#212*, *#211*, *#210*, *#209*, *#208*, *#207*, *#206*, and *#203*. Section *Tasks* also presents a summary with the total numbers of detected agile smells in the tasks (9 agile smells occurrences and 10 preconditions not met cases).

Figure 8.6 shows an expanded view of section *Tasks*. According to this snippet, *AS 06* was detected in task *#202* because the task complexity is higher than the maximum allowed complexity defined in the agile smell specification. For tasks *#192*, *#191*, *#190*, and *#189*, the preconditions to detect *AS 06* are not met because these tasks have no defined effort. The remaining tasks have no agile smells.

### 8.6.5 Phase 5: Reporting

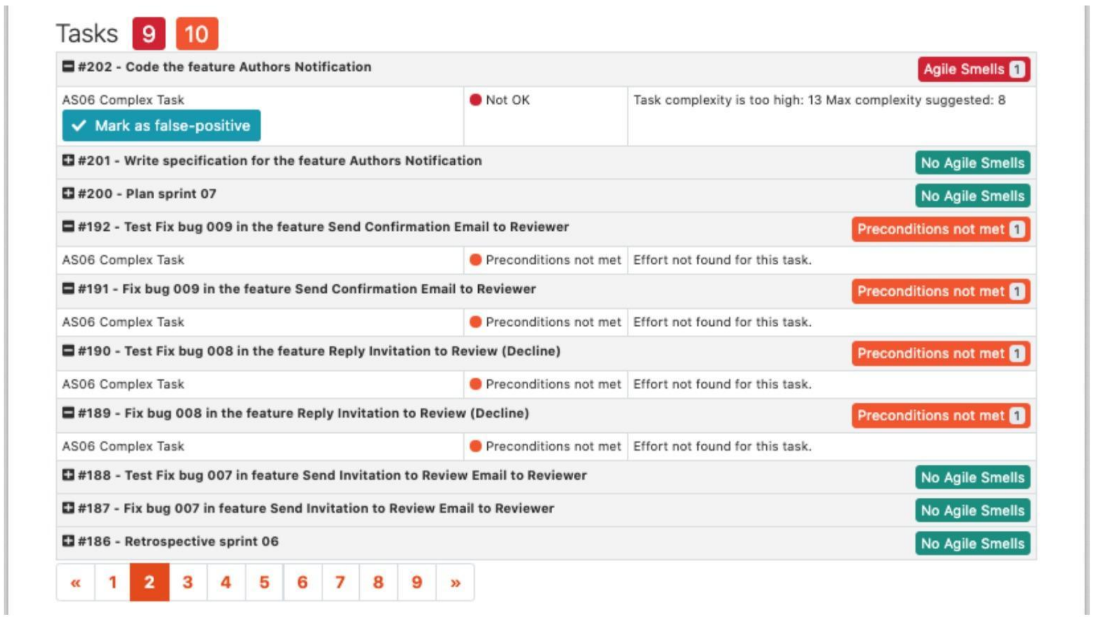
The *AgileQube App* found 27 agile smells in the *Agile Smells Report* which leads to a precision of 100 percent and a recall of 100 percent. Additionally, the *Agile Smells Report* indicate 17 situations where the agile smells were not applicable and 11 situa-



Iterations 18 17

| | | |
|---|---|---|
| **Iteration - Sprint 07 - Authors Notifications** | | Agile Smells 3 Not applicable 3 |
| AS03 Iteration Without A Deliverable ✓ Mark as false-positive | ● Not OK | No Deploy Activity found on the Iteration |
| AS04 Goals Not Defined or Poorly Defined ✓ Mark as false-positive | ● Not OK | Goals poorly defined for this Iteration. The goals should be described in the description section. |
| AS09 Iteration Started Without An Estimated Effort ✓ Mark as false-positive | ● Not OK | The Iteration is already open but the following Tasks has no estimated effort: #205 #204 |
| AS07 Iteration Without An Iteration Retrospective | ● Not Applicable | Iteration is not closed. |
| AS08 Absence of Timeboxed Iteration | ● Not Applicable | Iteration is not closed. |
| AS10 Iteration Without An Iteration Review | ● Not Applicable | Iteration is not closed. |
| **Iteration - Sprint 06 - Manuscript Review** | | Agile Smells 3 Not applicable 3 |
| AS01 Lower Priority Task Executed First ✓ Mark as false-positive | ● Not OK | Task #182 (doing - priority:normal) has priority lower than task #190 (todo - priority:high) |
| AS05 Iteration Without An Iteration Planning ✓ Mark as false-positive | ● Not OK | No Iteration Planning Activity found on the Iteration |
| AS09 Iteration Started Without An Estimated Effort ✓ Mark as false-positive | ● Not OK | The Iteration is already open but the following Tasks has no estimated effort: #192 #191 #190 #189 #182 |
| AS07 Iteration Without An Iteration Retrospective | ● Not Applicable | Iteration is not closed. |
| AS08 Absence of Timeboxed Iteration | ● Not Applicable | Iteration is not closed. |
| AS10 Iteration Without An Iteration Review | ● Not Applicable | Iteration is not closed. |
| **Iteration - Sprint 05 - Reply Invitation to Review** | | Agile Smells 3 Not applicable 3 |
| AS01 Lower Priority Task Executed First ✓ Mark as false-positive | ● Not OK | Task #157 (doing - priority:normal) has priority lower than task #156 (todo - priority:high) |
| AS04 Goals Not Defined or Poorly Defined ✓ Mark as false-positive | ● Not OK | Goals poorly defined for this Iteration. The goals should be described in the description section. |
| AS09 Iteration Started Without An Estimated Effort ✓ Mark as false-positive | ● Not OK | The Iteration is already open but the following Tasks has no estimated effort: #143 #141 #140 |
| AS07 Iteration Without An Iteration Retrospective | ● Not Applicable | Iteration is not closed. |
| AS08 Absence of Timeboxed Iteration | ● Not Applicable | Iteration is not closed. |
| AS10 Iteration Without An Iteration Review | ● Not Applicable | Iteration is not closed. |
| **Iteration - Sprint 04 - Invite Reviewers** | | Agile Smells 2 Not applicable 2 |
| AS03 Iteration Without A Deliverable ✓ Mark as false-positive | ● Not OK | No Deploy Activity found on the Iteration |
| AS10 Iteration Without An Iteration Review ✓ Mark as false-positive | ● Not OK | No Iteration Review Activity found on the Iteration |
| AS01 Lower Priority Task Executed First | ● Not Applicable | Iteration is not open. |
| AS09 Iteration Started Without An Estimated Effort | ● Not Applicable | Iteration is not open. |
| **Iteration - Sprint 03 - Manuscript Submission** | | Agile Smells 2 Not applicable 2 |
| AS07 Iteration Without An Iteration Retrospective ✓ Mark as false-positive | ● Not OK | No Iteration Retrospective Activity found on the Iteration |
| AS08 Absence of Timeboxed Iteration ✓ Mark as false-positive | ● Not OK | Iteration is longer (19 days) than the maximun allowed duration (17 days) |
| AS01 Lower Priority Task Executed First | ● Not Applicable | Iteration is not open. |
| AS09 Iteration Started Without An Estimated Effort | ● Not Applicable | Iteration is not open. |
| **Iteration - Sprint 02 - Authors Registration** | | Agile Smells 2 Not applicable 2 |
| AS04 Goals Not Defined or Poorly Defined | ● Not OK | No goals defined for this Iteration. The goals should be described |

Figure 8.5: *Agile Smells Report* of the JSS project - *Iterations* section


Tasks 9 10

#202 - Code the feature Authors Notification — Agile Smells 1

| | | |
|---|---|---|
| AS06 Complex Task<br>✓ Mark as false-positive | ● Not OK | Task complexity is too high: 13 Max complexity suggested: 8 |

#201 - Write specification for the feature Authors Notification — No Agile Smells

#200 - Plan sprint 07 — No Agile Smells

#192 - Test Fix bug 009 in the feature Send Confirmation Email to Reviewer — Preconditions not met 1

| | | |
|---|---|---|
| AS06 Complex Task | ● Preconditions not met | Effort not found for this task. |

#191 - Fix bug 009 in the feature Send Confirmation Email to Reviewer — Preconditions not met 1

| | | |
|---|---|---|
| AS06 Complex Task | ● Preconditions not met | Effort not found for this task. |

#190 - Test Fix bug 008 in the feature Reply Invitation to Review (Decline) — Preconditions not met 1

| | | |
|---|---|---|
| AS06 Complex Task | ● Preconditions not met | Effort not found for this task. |

#189 - Fix bug 008 in the feature Reply Invitation to Review (Decline) — Preconditions not met 1

| | | |
|---|---|---|
| AS06 Complex Task | ● Preconditions not met | Effort not found for this task. |

#188 - Test Fix bug 007 in feature Send Invitation to Review Email to Reviewer — No Agile Smells

#187 - Fix bug 007 in feature Send Invitation to Review Email to Reviewer — No Agile Smells

#186 - Retrospective sprint 06 — No Agile Smells

« 1 2 3 4 5 6 7 8 9 »

Figure 8.6: *Agile Smells Report* of the JSS project - *Tasks* section

tion where the preconditions to detected the agile smells were not met. The average computational time for the generation of the *Agile Smells Report (preliminary)* was around 15 seconds.

The high recall and precision observed in this experiment revealed that, in this scenario, the *AgileQube App* was able to detect all known agile smells and that all detected agile smells were indeed an agile smell. Such efficiency and effectiveness is explained by the size and controlled nature of the experiment. The project structure (iterations and tasks) was relatively small and the agile smells were deliberately injected in this structure. That was no surprise the lack of false-positive results in such a simulated dataset. However, since this is a controlled experiment, these results may not represent a solid conclusion and they reveal that more empirical experiments may be needed to further investigate the ability of the proposed approach to detect agile smells in agile projects.

To mitigate this threat, we conducted an additional case study using a private and real agile project from a medium-sized software company.

## 8.7 *Terminal Operational System*

The fourth strategy to validate the *AgileQube Approach* was made with a private and real agile project. Given the limitations in the results observed in the case study with a simulated project, the execution of a case study with a private and real project was important to provide us with further experimental evidence to validate our approach.

### 8.7.1 Phase 1: Case study design

#### Case Study Objective, Research Question, and validation criteria

This case study has the same objective, research question and validation criteria presented in Section 8.6.1.

#### Case selection

The selection of the *private and real agile project* considered the following criteria: *(a)* the project should be a conventional project (not an open-source project); *(b)* the project should have a well defined team; *(c)* the project should have a well defined scope; *(d)* the project should have a project manager (or an equivalent role); *(e)* the project should work through a well defined plan (with tasks and iterations); *(f)* preferably conducted in a software company; and *(g)* none of the authors of this research must have participated in the project.

Selecting such a project was particularly difficult because most of the companies that our research group usually cooperate with were facing an unprecedented situation. The new social-distance policies had prevented us, for example, from conducting face-to-face experiments to observe the occurrence of agile smells *in loco*. Additionally, the feedback we have received from some companies suggested the challenges of working in a fully virtual environment have made development teams less available for academic experiments.

Despite these limitations, we were able to select a project from a medium-sized software development organization to use as case study. The selected project aims the development of a Terminal Operating System (TOS) for a large logistic company. The TOS had about 8,000 tasks distributed over 125 iterations when we performed the experiment.

#### Project Structure Preparation

The project structure preparation consisted in selecting a subset of iterations and tasks to be subject of validation, extracting the selected project structure from the proprietary project management system, and inputting the extracted project structure into a *ZenHub* project created for this experiment. Two experienced members of the project were assigned to help us in the project structure preparation steps.

The members have been working in this project for more than 5 years and have graduate degree in computer science.

The selected subset project structure had a total of 279 tasks distributed over 6 iterations:

1. *Sprint 102*: 44 tasks/26 calendar days/18 business days;
2. *Sprint 103*: 46 tasks/33 calendar days/20 business days;
3. *Sprint 117*: 28 tasks/15 calendar days/11 business days;
4. *Sprint 118*: 30 tasks/16 calendar days/11 business days;
5. *Sprint 119*: 42 tasks/16 calendar days/12 business days;
6. *Sprint 120*: 89 tasks/23 calendar days/15 business days.

The 6 iterations and the 279 tasks that compose the dataset selected to this case study are shown in Table G.1 of Appendix G.

Then, we entered the extracted project structure into the *GitHub*[6] and *ZenHub*[7] projects that were created for this experiment.

#### Known Agile Smells

In order to calculate the precision and recall measures, we needed to know the real agile smells present in the selected project structure. To identify occurrences of the agile smells *AS 01* to *AS 10*, we, along with the assigned team members, manually analyzed the selected dataset. When in doubt, we referred to the *Catalogue of Agile Smells* to decide whether the issue was actually an agile smell. The 61 agile smells manually identified in this phase are presented in Table 8.2.

### 8.7.2 Phase 2: Preparation for data collection

The preparation for data collection in this case study consists of indicating to the *ETL Module* how to load the project data from *ZenHub*. The project configuration, that is shown in Figure 8.7, contains the following information:

(A) Github and Zenhub identifiers and API access tokens;

(B) Priorities Map;

(C) Status Map; and

(D) Process Activities Map.

[6]`https://github.com/utelemaco/tos-case-study-agileqube/issues`

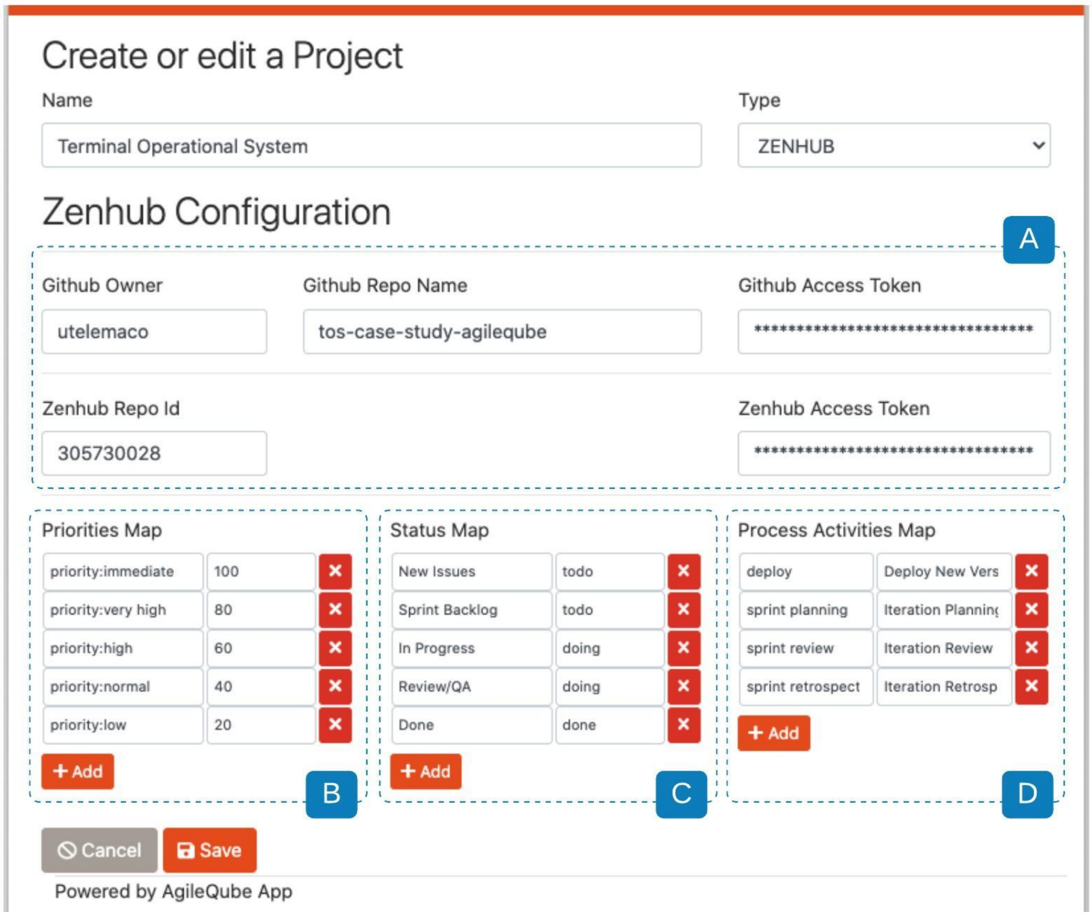
[7]`https://app.zenhub.com/workspaces/tos-5f8faf96ca3c9b00160b8d1f/board?repos=305730028`

Table 8.2: Agile smells manually identified in the TOS project

| **Agile Smell** | **Target** | **Occurrences** |
|---|---|---|
| (AS 01) *Lower Priority Tasks Executed First* | Sprint 120 | 1 |
| (AS 02) *Absence of Frequent Deliveries* | Project | 1 |
| (AS 03) *Iteration Without a Deliverable* | Sprint 117<br>Sprint 118<br>Sprint 119 | 3 |
| (AS 04) *Goals Not Defined or Poorly Defined* | Sprint 102<br>Sprint 117<br>Sprint 118<br>Sprint 119<br>Sprint 120 | 5 |
| (AS 05) *Iteration Without an Iteration Planning* | | 0 |
| (AS 06) *Complex Tasks* | #27780 #27973 #29932<br>#29953 #29557 #30042<br>#30045 #30046 #30048<br>#30035 #29930 #29927<br>#30051 #30136 #30175<br>#30180 #30187 #30192<br>#30196 #30202 #30203<br>#30205 #30207 #30208<br>#30209 #30218 #30233<br>#30235 #30236 #30238<br>#30241 #30242 #30244<br>#30479 #30361 #30442 | 36 |
| (AS 07) *Iteration Without an Iteration Retrospective* | Sprint 102<br>Sprint 103<br>Sprint 117<br>Sprint 119 | 4 |
| (AS 08) *Absence of Timeboxed Iteration* | Sprint 102<br>Sprint 103<br>Sprint 117<br>Sprint 120 | 4 |
| (AS 09) *Iteration Started without an Estimated Effort* | Sprint 117<br>Sprint 118<br>Sprint 120 | 3 |
| (AS 10) *Iteration Without an Iteration Review* | Sprint 103<br>Sprint 117<br>Sprint 118<br>Sprint 119 | 4 |
| | **Total** | **61** |

There are 5 labels to represent task priorities (*priority:immediate*, *priority:very high*, *priority:high*, *priority:normal*, and *priority:low*), four status (*New Issues* and *Sprint Backlog* that map to *todo*, *In Progress* that maps to *doing*, and *Done*) and four labels to represent the activities from the software process (*deploy* that corresponds to the activity *Deploy New version*, *sprint planning* that corresponds to the activity *Iteration Planning*, *sprint review* that corresponds to *Iteration Review*, and *sprint retrospective* that corresponds to *Iteration Retrospective*).

This configuration enables the *ETL Module* to load project data from *ZenHub*, i.e. to access the platform, extract the data, transform the extracted data and send the transformed data to the detection engine as described in Section 7.3.2.



Create or edit a Project

Name: Terminal Operational System

Type: ZENHUB

Zenhub Configuration

A

Github Owner: utelemaco

Github Repo Name: tos-case-study-agileqube

Github Access Token: ************************************

Zenhub Repo Id: 305730028

Zenhub Access Token: ************************************

Priorities Map

| | |
|---|---|
| priority:immediate | 100 |
| priority:very high | 80 |
| priority:high | 60 |
| priority:normal | 40 |
| priority:low | 20 |

B

Status Map

| | |
|---|---|
| New Issues | todo |
| Sprint Backlog | todo |
| In Progress | doing |
| Review/QA | doing |
| Done | done |

C

Process Activities Map

| | |
|---|---|
| deploy | Deploy New Vers |
| sprint planning | Iteration Plannin |
| sprint review | Iteration Review |
| sprint retrospect | Iteration Retrosp |

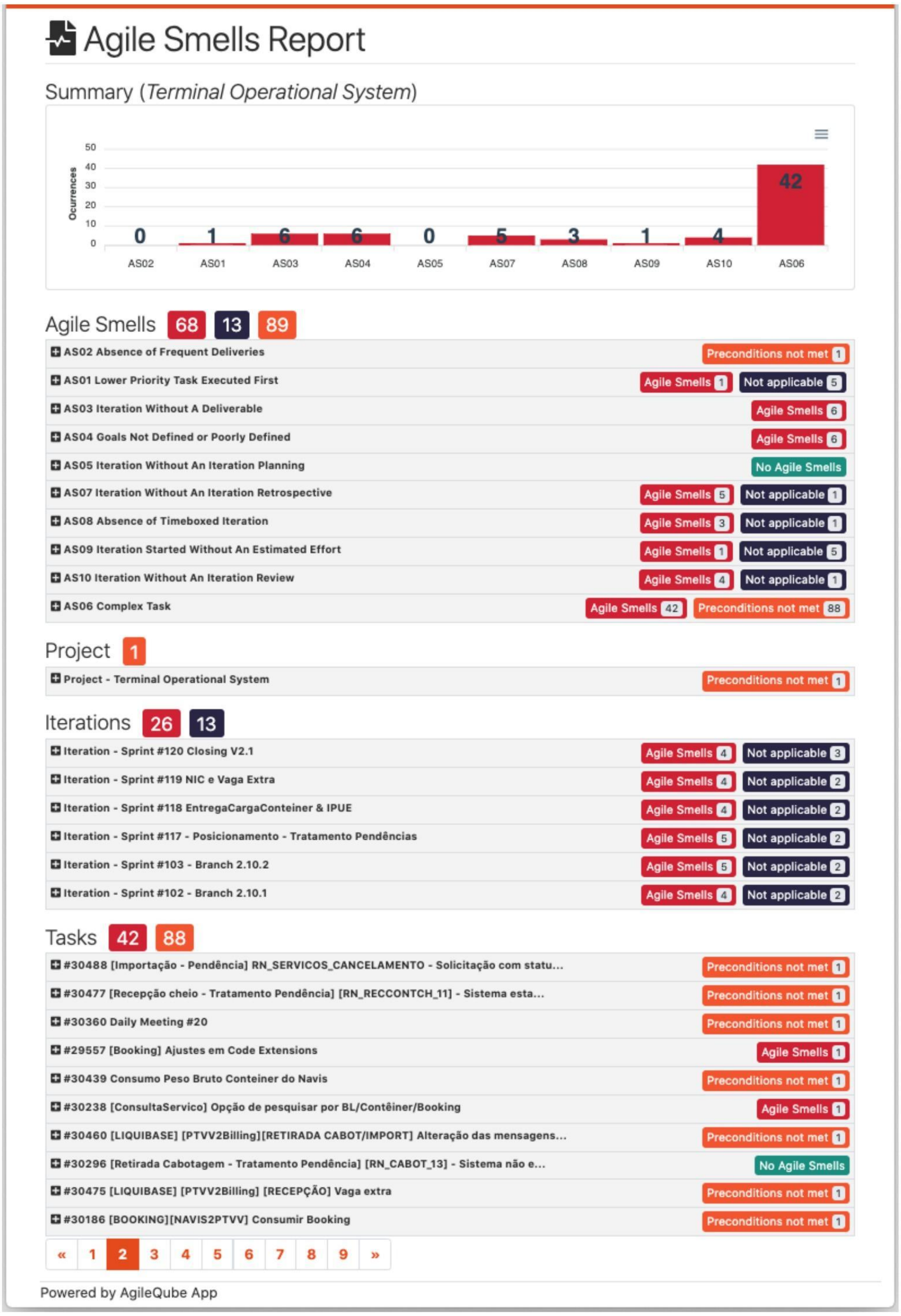
D

Figure 8.7: TOS project configuration in the *AgileQube App*

## 8.7.3 Phase 3: Data Collection

After configuring the TOS project, the *AgileQube App* was able to load the project data from *ZenHub*, detect the agile smells and generate an *Agile Smells Report*.

Figure 8.8 shows an overview of the generated report that can be divided in 5 sections: *Summary*, *Agile Smells*, *Project*, *Iterations*, and *Tasks*.



Agile Smells Report

Summary (*Terminal Operational System*)

Agile Smells 68 13 89

|  |  |
|---|---|
| AS02 Absence of Frequent Deliveries | Preconditions not met 1 |
| AS01 Lower Priority Task Executed First | Agile Smells 1 · Not applicable 5 |
| AS03 Iteration Without A Deliverable | Agile Smells 6 |
| AS04 Goals Not Defined or Poorly Defined | Agile Smells 6 |
| AS05 Iteration Without An Iteration Planning | No Agile Smells |
| AS07 Iteration Without An Iteration Retrospective | Agile Smells 5 · Not applicable 1 |
| AS08 Absence of Timeboxed Iteration | Agile Smells 3 · Not applicable 1 |
| AS09 Iteration Started Without An Estimated Effort | Agile Smells 1 · Not applicable 5 |
| AS10 Iteration Without An Iteration Review | Agile Smells 4 · Not applicable 1 |
| AS06 Complex Task | Agile Smells 42 · Preconditions not met 88 |

Project 1

|  |  |
|---|---|
| Project - Terminal Operational System | Preconditions not met 1 |

Iterations 26 13

|  |  |
|---|---|
| Iteration - Sprint #120 Closing V2.1 | Agile Smells 4 · Not applicable 3 |
| Iteration - Sprint #119 NIC e Vaga Extra | Agile Smells 4 · Not applicable 2 |
| Iteration - Sprint #118 EntregaCargaConteiner & IPUE | Agile Smells 4 · Not applicable 2 |
| Iteration - Sprint #117 - Posicionamento - Tratamento Pendências | Agile Smells 5 · Not applicable 2 |
| Iteration - Sprint #103 - Branch 2.10.2 | Agile Smells 5 · Not applicable 2 |
| Iteration - Sprint #102 - Branch 2.10.1 | Agile Smells 4 · Not applicable 2 |

Tasks 42 88

|  |  |
|---|---|
| #30488 [Importação - Pendência] RN_SERVICOS_CANCELAMENTO - Solicitação com statu... | Preconditions not met 1 |
| #30477 [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_11] - Sistema esta... | Preconditions not met 1 |
| #30360 Daily Meeting #20 | Preconditions not met 1 |
| #29557 [Booking] Ajustes em Code Extensions | Agile Smells 1 |
| #30439 Consumo Peso Bruto Conteiner do Navis | Preconditions not met 1 |
| #30238 [ConsultaServico] Opção de pesquisar por BL/Contêiner/Booking | Agile Smells 1 |
| #30460 [LIQUIBASE] [PTVV2Billing][RETIRADA CABOT/IMPORT] Alteração das mensagens... | Preconditions not met 1 |
| #30296 [Retirada Cabotagem - Tratamento Pendência] [RN_CABOT_13] - Sistema não e... | No Agile Smells |
| #30475 [LIQUIBASE] [PTVV2Billing] [RECEPÇÃO] Vaga extra | Preconditions not met 1 |
| #30186 [BOOKING][NAVIS2PTVV] Consumir Booking | Preconditions not met 1 |

« 1 2 3 4 5 6 7 8 9 »

Powered by AgileQube App

Figure 8.8: *Agile Smells Report* overview of TOS project

### 8.7.4 Phase 4: Analysis of collected data

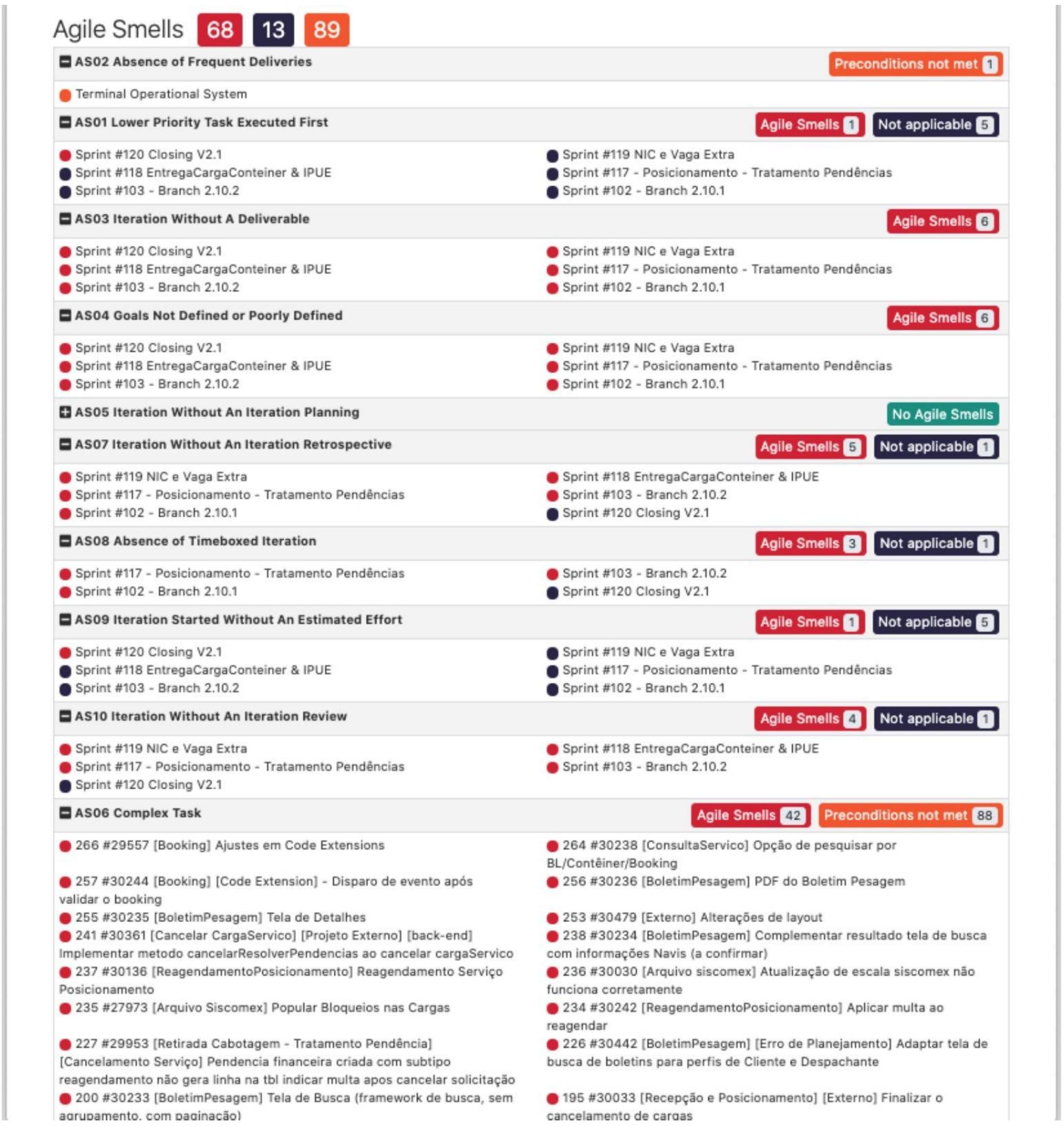
The bar chart in the section *Summary* presents the distribution of agile smells occurrences over the project and reveals that: *(a)* the agile smell *AS 06* has the highest number of occurrences (42 in total); *(b)* the agile smells *AS 03 Iteration Without a Deliverable* and *AS 04 Goals Not Defined or Poorly Defined* have 6 occurrences each; *(c)* the agile smell *AS 07 Iteration Without an Iteration Retrospective* has 5 occurrences; *(d)* the agile smell *AS 10 Iteration Without an Iteration Review* has 4 occurrences; *(e)* the agile smell *AS 08 Absence of Timeboxed Iteration* has 3 occurrences; *(f)* the agile smells *AS 01 Lower Priority Tasks Executed First* and *AS 09 Iteration Started without an Estimated Effort* have 6 occurrences each; *(g)* the agile smells *AS 02 Absence of Frequent Deliveries* and *AS 05 Iteration Without an Iteration Planning* have no occurrences.

Section *Agile Smells* presents the 10 agile smells and their corresponding occurrences. For example, the agile smell *AS 01 Lower Priority Tasks Executed First* has 1 occurrence and 5 not applicable cases. The agile smell *AS 05 Iteration Without an Iteration Planning* has no agile smell. The agile smell *AS 07 Iteration Without an Iteration Retrospective* has 5 occurrences and 1 not applicable case. The summary presented in the section title reveals: 68 agile smells occurrences, 13 not applicable cases, and 89 cases where the preconditions for detection were not met.

An expanded view of section *Agile Smells* is shown in Figure 8.9. In the expanded view, it is possible to check the target elements associated with each agile smell. For example, the agile smell *AS 01 Lower Priority Tasks Executed First* is detected in the iteration *Sprint 120* but is considered not applicable for the iterations *Sprint 119*, *Sprint 118*, *Sprint 117*, *Sprint 103*, and *Sprint 102*. The agile smell *AS 08 Absence of Timeboxed Iteration* is detected in the iterations *Sprint 117*, *Sprint 103*, and *Sprint 102* but is considered not applicable for the iteration *Sprint 120*.

Section *Project* presents the agile smells whose target element is the *Project*. Only the agile smell *AS 02 Absence of Frequent Deliveries* appears in this category. An expanded view of the section *Project* (see Figure 8.10) reveals the preconditions to detect the agile smell *AS 02* were not met since there is no delivery interval information for the given project.

Section *Iterations* presents the 6 iterations and their corresponding agile smells. For each iteration, it is shown labels indicating: the number of agile smells (*Agile Smells*), the number of not applicable cases (*Not Applicable*), the number of cases where the preconditions to detect the agile smell were not met (*Preconditions not*


Agile Smells 68 13 89

**AS02 Absence of Frequent Deliveries** — Preconditions not met 1

🟠 Terminal Operational System

**AS01 Lower Priority Task Executed First** — Agile Smells 1 | Not applicable 5

🔴 Sprint #120 Closing V2.1
⚫ Sprint #118 EntregaCargaConteiner & IPUE
⚫ Sprint #103 - Branch 2.10.2
⚫ Sprint #119 NIC e Vaga Extra
⚫ Sprint #117 - Posicionamento - Tratamento Pendências
⚫ Sprint #102 - Branch 2.10.1

**AS03 Iteration Without A Deliverable** — Agile Smells 6

🔴 Sprint #120 Closing V2.1
🔴 Sprint #118 EntregaCargaConteiner & IPUE
🔴 Sprint #103 - Branch 2.10.2
🔴 Sprint #119 NIC e Vaga Extra
🔴 Sprint #117 - Posicionamento - Tratamento Pendências
🔴 Sprint #102 - Branch 2.10.1

**AS04 Goals Not Defined or Poorly Defined** — Agile Smells 6

🔴 Sprint #120 Closing V2.1
🔴 Sprint #118 EntregaCargaConteiner & IPUE
🔴 Sprint #103 - Branch 2.10.2
🔴 Sprint #119 NIC e Vaga Extra
🔴 Sprint #117 - Posicionamento - Tratamento Pendências
🔴 Sprint #102 - Branch 2.10.1

**AS05 Iteration Without An Iteration Planning** — No Agile Smells

**AS07 Iteration Without An Iteration Retrospective** — Agile Smells 5 | Not applicable 1

🔴 Sprint #119 NIC e Vaga Extra
🔴 Sprint #117 - Posicionamento - Tratamento Pendências
🔴 Sprint #102 - Branch 2.10.1
🔴 Sprint #118 EntregaCargaConteiner & IPUE
🔴 Sprint #103 - Branch 2.10.2
⚫ Sprint #120 Closing V2.1

**AS08 Absence of Timeboxed Iteration** — Agile Smells 3 | Not applicable 1

🔴 Sprint #117 - Posicionamento - Tratamento Pendências
🔴 Sprint #102 - Branch 2.10.1
🔴 Sprint #103 - Branch 2.10.2
⚫ Sprint #120 Closing V2.1

**AS09 Iteration Started Without An Estimated Effort** — Agile Smells 1 | Not applicable 5

🔴 Sprint #120 Closing V2.1
⚫ Sprint #118 EntregaCargaConteiner & IPUE
⚫ Sprint #103 - Branch 2.10.2
⚫ Sprint #119 NIC e Vaga Extra
⚫ Sprint #117 - Posicionamento - Tratamento Pendências
⚫ Sprint #102 - Branch 2.10.1

**AS10 Iteration Without An Iteration Review** — Agile Smells 4 | Not applicable 1

🔴 Sprint #119 NIC e Vaga Extra
🔴 Sprint #117 - Posicionamento - Tratamento Pendências
⚫ Sprint #120 Closing V2.1
🔴 Sprint #118 EntregaCargaConteiner & IPUE
🔴 Sprint #103 - Branch 2.10.2

**AS06 Complex Task** — Agile Smells 42 | Preconditions not met 88

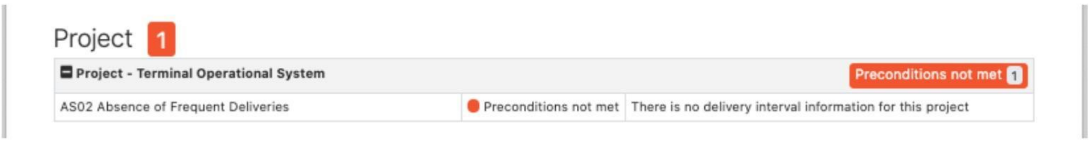
🔴 266 #29557 [Booking] Ajustes em Code Extensions
🔴 264 #30238 [ConsultaServico] Opção de pesquisar por BL/Contêiner/Booking
🔴 257 #30244 [Booking] [Code Extension] - Disparo de evento após validar o booking
🔴 256 #30236 [BoletimPesagem] PDF do Boletim Pesagem
🔴 255 #30235 [BoletimPesagem] Tela de Detalhes
🔴 253 #30479 [Externo] Alterações de layout
🔴 241 #30361 [Cancelar CargaServico] [Projeto Externo] [back-end] Implementar metodo cancelarResolverPendencias ao cancelar cargaServico
🔴 238 #30234 [BoletimPesagem] Complementar resultado tela de busca com informações Navis (a confirmar)
🔴 237 #30136 [ReagendamentoPosicionamento] Reagendamento Serviço Posicionamento
🔴 236 #30030 [Arquivo siscomex] Atualização de escala siscomex não funciona corretamente
🔴 235 #27973 [Arquivo Siscomex] Popular Bloqueios nas Cargas
🔴 234 #30242 [ReagendamentoPosicionamento] Aplicar multa ao reagendar
🔴 227 #29953 [Retirada Cabotagem - Tratamento Pendência] [Cancelamento Serviço] Pendencia financeira criada com subtipo reagendamento não gera linha na tbl indicar multa apos cancelar solicitação
🔴 226 #30442 [BoletimPesagem] [Erro de Planejamento] Adaptar tela de busca de boletins para perfis de Cliente e Despachante
🔴 200 #30233 [BoletimPesagem] Tela de Busca (framework de busca, sem agrupamento, com paginação)
🔴 195 #30033 [Recepção e Posicionamento] [Externo] Finalizar o cancelamento de cargas

Figure 8.9: *Agile Smells Report* of TOS project - *Agile Smells* section

*met*) or a label indicating all previous numbers are zero (*No Agile Smells*). For example, iteration *Sprint 120* has 4 agile smells and 3 not applicable cases. Iteration *Sprint 117* has 5 agile smells and 2 not applicable cases. Section *Iterations* also shows a summary with the total numbers of detected agile smells in the iterations (26 agile smells occurrences and 13 not applicable cases).

An expanded view of section *Iterations* is shown in Figure 8.11.

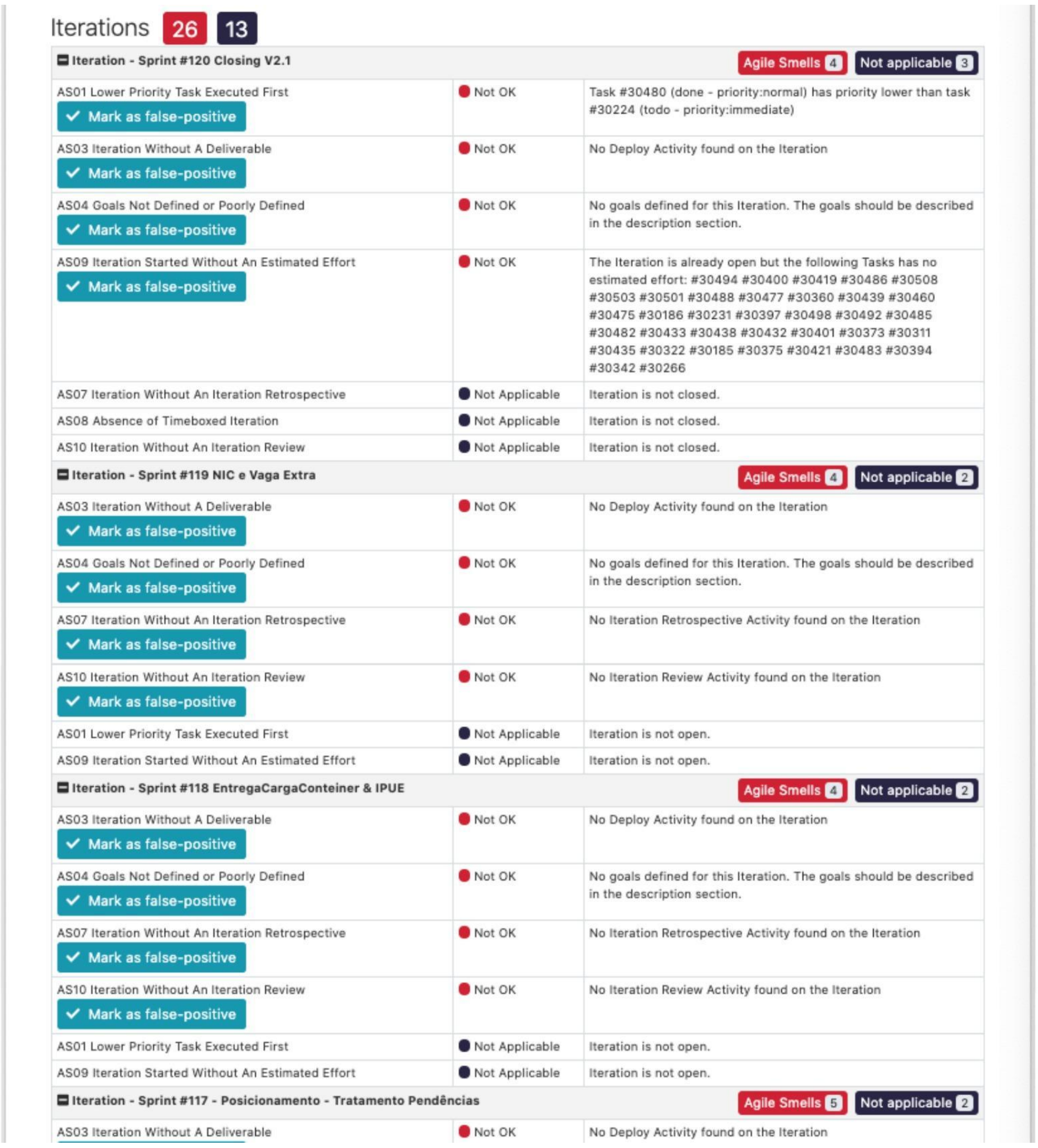
In the expanded view, it is possible to see details of the agile smell occurrences such as: agile smell name, result and description. For example, the 4 agile smells detected in iteration *Sprint 120* are *AS 01*, *AS 03*, *AS 04*, and *AS 09*. *AS 01 Lower Priority Tasks Executed First* was detected because a normal priority task (*#30480*) was executed before a high priority task (*#30224*). *AS 03 Iteration*



Project 1

| Project - Terminal Operational System | | Preconditions not met 1 |
|---|---|---|
| AS02 Absence of Frequent Deliveries | ● Preconditions not met | There is no delivery interval information for this project |

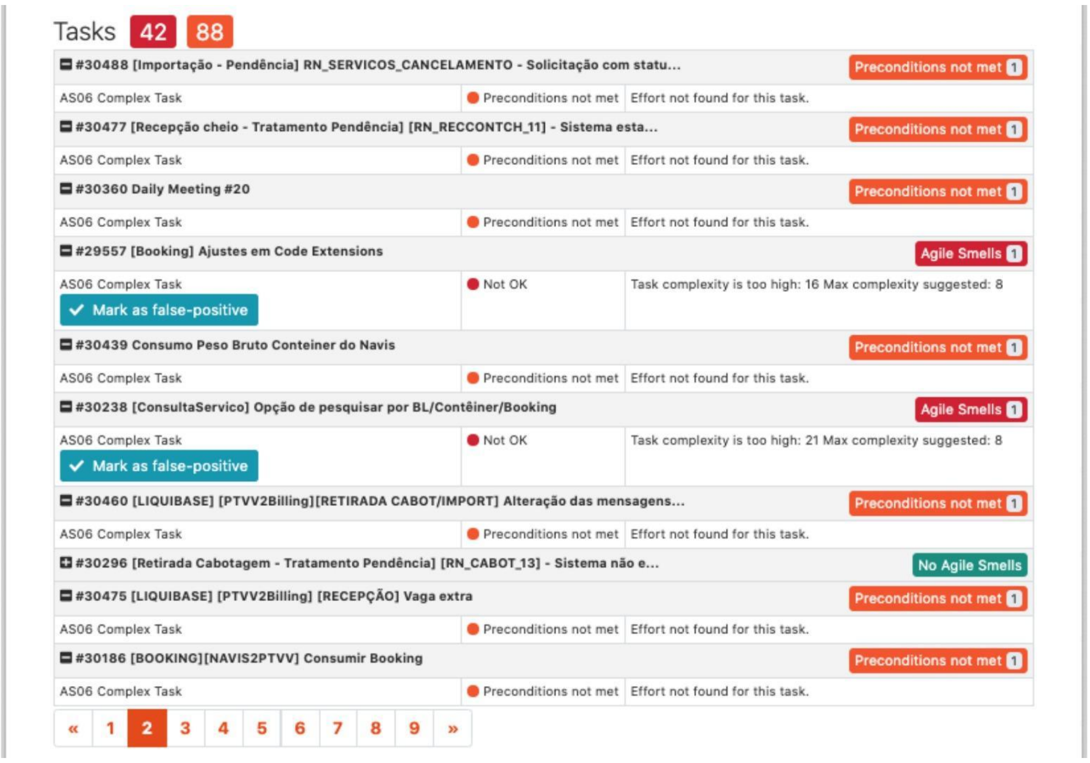
Figure 8.10: *Agile Smells Report* of TOS project - *Project* section



Iterations 26 13

| Iteration - Sprint #120 Closing V2.1 | | Agile Smells 4 Not applicable 3 |
|---|---|---|
| AS01 Lower Priority Task Executed First ✓ Mark as false-positive | ● Not OK | Task #30480 (done - priority:normal) has priority lower than task #30224 (todo - priority:immediate) |
| AS03 Iteration Without A Deliverable ✓ Mark as false-positive | ● Not OK | No Deploy Activity found on the Iteration |
| AS04 Goals Not Defined or Poorly Defined ✓ Mark as false-positive | ● Not OK | No goals defined for this Iteration. The goals should be described in the description section. |
| AS09 Iteration Started Without An Estimated Effort ✓ Mark as false-positive | ● Not OK | The Iteration is already open but the following Tasks has no estimated effort: #30494 #30400 #30419 #30486 #30508 #30503 #30501 #30488 #30477 #30360 #30439 #30460 #30475 #30186 #30231 #30397 #30498 #30492 #30485 #30482 #30433 #30438 #30432 #30401 #30373 #30311 #30435 #30322 #30185 #30375 #30421 #30483 #30394 #30342 #30266 |
| AS07 Iteration Without An Iteration Retrospective | ● Not Applicable | Iteration is not closed. |
| AS08 Absence of Timeboxed Iteration | ● Not Applicable | Iteration is not closed. |
| AS10 Iteration Without An Iteration Review | ● Not Applicable | Iteration is not closed. |
| **Iteration - Sprint #119 NIC e Vaga Extra** | | Agile Smells 4 Not applicable 2 |
| AS03 Iteration Without A Deliverable ✓ Mark as false-positive | ● Not OK | No Deploy Activity found on the Iteration |
| AS04 Goals Not Defined or Poorly Defined ✓ Mark as false-positive | ● Not OK | No goals defined for this Iteration. The goals should be described in the description section. |
| AS07 Iteration Without An Iteration Retrospective ✓ Mark as false-positive | ● Not OK | No Iteration Retrospective Activity found on the Iteration |
| AS10 Iteration Without An Iteration Review ✓ Mark as false-positive | ● Not OK | No Iteration Review Activity found on the Iteration |
| AS01 Lower Priority Task Executed First | ● Not Applicable | Iteration is not open. |
| AS09 Iteration Started Without An Estimated Effort | ● Not Applicable | Iteration is not open. |
| **Iteration - Sprint #118 EntregaCargaConteiner & IPUE** | | Agile Smells 4 Not applicable 2 |
| AS03 Iteration Without A Deliverable ✓ Mark as false-positive | ● Not OK | No Deploy Activity found on the Iteration |
| AS04 Goals Not Defined or Poorly Defined ✓ Mark as false-positive | ● Not OK | No goals defined for this Iteration. The goals should be described in the description section. |
| AS07 Iteration Without An Iteration Retrospective ✓ Mark as false-positive | ● Not OK | No Iteration Retrospective Activity found on the Iteration |
| AS10 Iteration Without An Iteration Review ✓ Mark as false-positive | ● Not OK | No Iteration Review Activity found on the Iteration |
| AS01 Lower Priority Task Executed First | ● Not Applicable | Iteration is not open. |
| AS09 Iteration Started Without An Estimated Effort | ● Not Applicable | Iteration is not open. |
| **Iteration - Sprint #117 - Posicionamento - Tratamento Pendências** | | Agile Smells 5 Not applicable 2 |
| AS03 Iteration Without A Deliverable | ● Not OK | No Deploy Activity found on the Iteration |

Figure 8.11: *Agile Smells Report* of TOS project - *Iterations* section


Tasks 42 88

| | | |
|---|---|---|
| **#30488 [Importação - Pendência] RN_SERVICOS_CANCELAMENTO - Solicitação com statu...** | | Preconditions not met 1 |
| AS06 Complex Task | ● Preconditions not met | Effort not found for this task. |
| **#30477 [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_11] - Sistema esta...** | | Preconditions not met 1 |
| AS06 Complex Task | ● Preconditions not met | Effort not found for this task. |
| **#30360 Daily Meeting #20** | | Preconditions not met 1 |
| AS06 Complex Task | ● Preconditions not met | Effort not found for this task. |
| **#29557 [Booking] Ajustes em Code Extensions** | | Agile Smells 1 |
| AS06 Complex Task ✓ Mark as false-positive | ● Not OK | Task complexity is too high: 16 Max complexity suggested: 8 |
| **#30439 Consumo Peso Bruto Conteiner do Navis** | | Preconditions not met 1 |
| AS06 Complex Task | ● Preconditions not met | Effort not found for this task. |
| **#30238 [ConsultaServico] Opção de pesquisar por BL/Contêiner/Booking** | | Agile Smells 1 |
| AS06 Complex Task ✓ Mark as false-positive | ● Not OK | Task complexity is too high: 21 Max complexity suggested: 8 |
| **#30460 [LIQUIBASE] [PTVV2Billing][RETIRADA CABOT/IMPORT] Alteração das mensagens...** | | Preconditions not met 1 |
| AS06 Complex Task | ● Preconditions not met | Effort not found for this task. |
| **#30296 [Retirada Cabotagem - Tratamento Pendência] [RN_CABOT_13] - Sistema não e...** | | No Agile Smells |
| **#30475 [LIQUIBASE] [PTVV2Billing] [RECEPÇÃO] Vaga extra** | | Preconditions not met 1 |
| AS06 Complex Task | ● Preconditions not met | Effort not found for this task. |
| **#30186 [BOOKING][NAVIS2PTVV] Consumir Booking** | | Preconditions not met 1 |
| AS06 Complex Task | ● Preconditions not met | Effort not found for this task. |

« 1 2 3 4 5 6 7 8 9 »

Figure 8.12: *Agile Smells Report* of TOS project - *Tasks* section

*Without a Deliverable* was detected because no deployment activity was found in the iteration. *AS 04 Goals Not Defined or Poorly Defined* was detected because no goal was defined to the iteration. *AS 09 Iteration Started without an Estimated Effort* was detected because the iteration is open but has 35 not estimated tasks. The agile smells *AS 07*, *AS 08*, and *AS 10* are not applicable for *Sprint 120* because the iteration is not closed.

Section *Tasks* lists the 279 tasks and their corresponding agile smells. Similarly to section *Iterations*, this section shows, for each task, labels indicating the number of agile smells associated with the task (*Agile Smells*, *Not applicable*, *Preconditions not met*, *No Agile Smell*). A pagination component is enabled when the list of tasks reaches a specific number of elements. According to the expanded view of the section *Tasks* shown in Figure 8.12, the agile smell *AS 06 Complex Tasks* was detected in the tasks *#29557* and *#30238* because their complexities (16 and 21 respectively) are higher than the suggested complexity defined in the agile smell specification (which is 8). For the tasks *#30488*, *#30477*, *#30360*, *#30439*, *#30460*, *#30475*, and *#30186*, the preconditions to detect the agile smell *AS 06 Complex Tasks* were not met because these tasks have no defined effort. No agile smells were detected in the task *#30296*. Section *Tasks* also presents a summary of the total numbers of detected agile smells in the tasks (42 agile smells occurrences and 88 preconditions not met cases).

### 8.7.5 Phase 5: Reporting

The *Agile Smells Report* of the *Terminal Operational System* project reveals the *AgileQube App* found 68 agile smells, 13 situations where the agile smells were not applicable and 89 situation where the preconditions to detected the agile smells were not met. The average computational time for the generation of the *Agile Smells Report (preliminary)* ranged between 10 and 15 seconds.

Table 8.3 shows a summary of the agile smells, the assessed targets (project, iterations and tasks) and their corresponding number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN). Where TP means the agile smell was correctly detected, FP means the agile smell was incorrectly detected, TN means the agile smell was correctly not detected, and FN means the agile smell was incorrectly not detected. For readability purposes, we do not include in the table target entities with a TN instance.

The precision and recall were calculated as follows:

*(a) 57 of the detected 68 agile smells were indeed an agile smell which leads to:*

$$\textbf{\textit{precision}} = \frac{\textbf{\textit{TP}}}{\textbf{\textit{(TP + FP)}}} = \frac{\textbf{\textit{57}}}{\textbf{\textit{68}}} = \textbf{\textit{83.8\%}}$$

*(b) 57 of 61 known agile smells were found which leads to:*

$$\textbf{\textit{recall}} = \frac{\textbf{\textit{TP}}}{\textbf{\textit{(TP + FN)}}} = \frac{\textbf{\textit{57}}}{\textbf{\textit{61}}} = \textbf{\textit{93.4\%}}$$

One of the limitations to assess the results observed in this case study is the lack of a framework or baseline to evaluate and compare the proposed approach with other agile smell detection tools. As the term *agile smell* and its use to support the assessment of agility are novelties proposed in this research, there are no other agile smell detection tool described in the literature. However, the results in this case study, although not conclusive, indicate that the proposed approach was capable of

Table 8.3: Agile smells in the TOS project (TP - True Positive; FP - False Positive; TN - True Negative; FN - False Negative)

| **Agile Smell** | **Target** | **TP** | **FP** | **TN** | **FN** |
|---|---|---|---|---|---|
| (AS 01) *Lower Priority Tasks Executed First* | Sprint 120 (TP) | 1 | 0 | 5 | 0 |
| (AS 02) *Absence of Frequent Deliveries* | Project (FN) | 0 | 0 | 0 | 1 |
| (AS 03) *Iteration Without a Deliverable* | Sprint 102 (FP)<br>Sprint 103 (FP)<br>Sprint 117 (TP)<br>Sprint 118 (TP)<br>Sprint 119 (TP)<br>Sprint 120 (FP) | 3 | 3 | 0 | 0 |
| (AS 04) *Goals Not Defined or Poorly Defined* | Sprint 102 (TP)<br>Sprint 103 (FP)<br>Sprint 117 (TP)<br>Sprint 118 (TP)<br>Sprint 119 (TP)<br>Sprint 120 (TP) | 5 | 1 | 0 | 0 |
| (AS 05) *Iteration Without an Iteration Planning* | | 0 | 0 | 6 | 0 |
| (AS 06) *Complex Tasks* | #14687 (FP) #27780 (TP)<br>#27973 (TP) #29927 (TP)<br>#29930 (TP) #29932 (TP)<br>#29953 (TP) #29557 (TP)<br>#30042 (TP) #30044 (FP)<br>#30045 (TP) #30046 (TP)<br>#30048 (TP) #30035 (TP)<br>#30030 (FP) #30033 (FP)<br>#30051 (TP) #30136 (TP)<br>#30175 (TP) #30180 (TP)<br>#30187 (TP) #30192 (TP)<br>#30196 (TP) #30202 (TP)<br>#30203 (TP) #30205 (TP)<br>#30207 (TP) #30208 (TP)<br>#30209 (TP) #30218 (TP)<br>#30219 (FP) #30233 (TP)<br>#30234 (FP) #30235 (TP)<br>#30236 (TP) #30238 (TP)<br>#30241 (TP) #30242 (TP)<br>#30244 (TP) #30479 (TP)<br>#30361 (TP) #30442 (TP) | 36 | 6 | 237 | 0 |
| (AS 07) *Iteration Without an Iteration Retrospective* | Sprint 102 (TP)<br>Sprint 103 (TP)<br>Sprint 117 (TP)<br>Sprint 118 (FP)<br>Sprint 119 (TP) | 4 | 1 | 1 | 0 |
| (AS 08) *Absence of Timeboxed Iteration* | Sprint 102 (TP)<br>Sprint 103 (TP)<br>Sprint 117 (TP)<br>Sprint 120 (FN) | 3 | 0 | 2 | 1 |
| (AS 09) *Iteration Started without an Estimated Effort* | Sprint 117 (FN)<br>Sprint 118 (FN)<br>Sprint 120 (TP) | 1 | 0 | 3 | 2 |
| (AS 10) *Iteration Without an Iteration Review* | Sprint 103 (TP)<br>Sprint 117 (TP)<br>Sprint 118 (TP)<br>Sprint 119 (TP) | 4 | 0 | 2 | 0 |
| | **Total** | **57** | **11** | **256** | **4** |

detecting agile smells in a real agile project with reasonable performance (about 15 seconds) and high precision and recall (83.8% and 93.4%).

## 8.8 Threats to validity

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

*Construct Validity.* This refers to what extent the experimental measures really represent what is investigated according to the research questions. A threat in this category regards the simulated project structure (i.e. the set of iterations and tasks) that were elaborated to the case study *Journal Submission System*. To mitigate the problem of building biased project structure, two independent Project Managers aided the definition of the simulated project structure.

Another threat to construct validity was the identification of "true" occurrences of agile smells that was necessary in the case study *Terminal Operational System* to calculate the precision and recall measures. To identify those agile smells, we had to manually analyze the project structure selected for the case study. To mitigate the bias of this strategy, two independent members of the project were assigned to help us in this step. Thus, we collaboratively identified the agile smells and when in doubt, we referred to the *Catalogue of Agile Smells* to decide together whether the issue was actually an agile smell.

*Conclusion Validity.* This aspect examines the extent to which conclusions derived using experimental results is valid. A threat in this category regards the number of case studies carried out in the research. Since we conducted a non-representative number of experiments, the results observed in the presented case studies are not conclusive. However, the results allowed us to observe that the proposed approach was capable of identifying agile smells in the selected agile projects with reasonable performance and high precision and recall.

## 8.9 Conclusion

In this chapter we presented the case studies that were conducted to validate the *AgileQube Approach*. We tried 4 strategies to demonstrate the use of the proposed approach: *(a)* OS projects hosted on *GitHub*; *(b)* OS projects hosted on *ZenHub*; *(c)* a simulated agile project (*Journal Submission System*); and *(d)* a real agile project (*Terminal Operational System*).

We started the demonstration and evaluation phase by exploring the use of OS projects (hosted on *GitHub* and *ZenHub*) to validate the proposed approach. The strategies involving OS projects would enable this research to reach a high number of projects and hence demonstrate the use of the proposed approach in different scenarios. However, these strategies revealed unsuccessful mostly because OS projects, unlike conventional agile projects, generally do not have a project plan, schedule or delivery list (MOCKUS *et al.* 2000 [212]) (that are fundamental data to detect agile smells).

In the strategy using a simulated agile project, we created a project hosted on *ZenHub*, that we named *Journal Submission System*, and, along with two independent Project Managers, defined its project structure (i.e., its iterations and tasks). We deliberately injected 27 agile smells into the JSS project structure. After configuring the project, the *AgileQube App* was capable of loading the project data from *ZenHub* and detecting all the injected agile smells. The approach achieved, in this case study, a precision and recall of 100 percent and the average computation time to load data from *ZenHub* and generate the *Agile Smells Report* was about 15 seconds.

In the strategy using a realistic agile project, we selected a project from a medium-size company, that we renamed to *Terminal Operational System*, and, along with two team members, selected a subset of the project structure to be used as dataset for the case study. We analyzed the selected project structure and manually detected 61 agile smells. Then we entered the project structure into the *GitHub* and *ZenHub* projects created for the case study. After configuring the project, the *AgileQube App* was capable of loading the project data from *ZenHub* and detecting 68 agile smells. The approach achieved, in this case study, a precision of 83.8 percent and recall of 93.4 percent. The average computation time to load data from *ZenHub* and generate the *Agile Smells Report* was about 15 seconds.

The results collected in the presented case studies, although not conclusive and not representative, represent a clear indication that the proposed approach was capable of detecting occurrences of the catalogued agile smells in the selected agile projects.

# Chapter 9

# Conclusion

*This chapter presents our final remarks about the presented thesis, including a summary of the research, our main contributions, threads to validity and future work.*

## 9.1 Summary

This research had as its main topic *agility assessment*, which is an assessment to reveal how a certain company has been adopting agile practices in its software development process. Despite being an important tool to assist in the adoption of agile development, the existing agility assessment solutions have some critical gaps.

In the early stage of this research, we investigated the existing agility assessment approaches and identified the following problems and limitations: P1 (*Unclear assessment criteria selection*); P2 (*Unclear assessment criteria representation*); P3 (*Lack of support for adding new assessment criterion*); P4 (*Manual data collection and input*); P5 (*Lack of real-time assessment feedback*); and P6 (*Limited Scalability*).

To address these problems, we extended the *code smell* term to the context of agility assessment and introduced the *agile smell* term to denote a situation that may impair the proper adoption of agile practices. Then, we proposed an agility assessment approach that automatically (or semi-automatically) detects agile smells in an agile project. To support the elaboration of such approach, we defined 4 specific goals: G1 (*Define a catalogue of agile smells*); G2 (*Define an agile smell representation approach*); G3 (*Define a data extraction approach*); and G4 (*Define and implement a computational system*).

In addition to the proposed approach, that we named *AgileQube Approach*, this research produced 4 contributions: the *Catalogue of Agile Smells*, the *Agile Project Metamodel*, the *Agile Smell Schema*, and the *AgileQube App*.

The *AgileQube Approach* is an agility assessment approach based on the DECOR method proposed by MOHA *et al.* 2009 [139]. While DECOR focuses on automatic detection of code smells in source code, the *AgileQube Approach* defines the steps and the components for specification and automatic detection of agile smells in agile projects. The approach has five phases: *Identification*, *Specification*, *Configuration*, *Detection*, and *Validation.* In the *Identification* phase, the agile smells that will be used in the following phases are selected. The *Specification* and *Configuration* phases focus on translating the descriptions of the selected agile smells into systematic specifications. These phases are supported by a metamodel (*Agile Project Metamodel*) and a specification language (*Agile Smell Schema*). In the *Detection* phase, the agile smells are ultimately detected by a detection engine and, in the last phase, *Validation*, these agile smells are manually analized and the false-positive occurrences discarded.

The *Catalogue of Agile Smells* was elaborated in the *Identification* phase of the approach in a three-step methodology: first, we conducted a literature review that analyzed peer-reviewed and grey literature studies to identify the agile smells, i.e, practices that may impair the adoption of an agile practice. Second, we conducted a survey with practitioners to reveal the relevance of these agile smells from an industry perspective. Finally, we organized the top 10 agile smells as a catalogue in a structure that provides a clear definition of the agile smells and indicates its name and description, which agile practices motivated the agile smell, and at least one identification strategy that guides the specification of the agile smell.

The role of the *Agile Project Metamodel* in the approach is twofold: *(a)* it is used to represent the data from the evaluated agile project and *(b)* it is used in the specification of the agile smells. The metamodel was also elaborated in three steps: first, we selected from the descriptions of the agile smells, the concepts necessary to represent an agile project. Second, we conducted a literature review to investigate how existing software development metamodels could be used to represent these elements. Third, we proposed a metamodel that combines elements from existing metamodels with new elements.

The *Agile Smell Schema*, along with the metamodel, enables the systematic specification of the agile smells into the approach. The schema is composed of 10 elements that provide a comprehensive definition of an agile smell. These elements can be divided into three categories: *(a)* elements to document the agile smell; *(b)* elements to relate the agile smell with the agile practices and methods that motivated the agile smell; and *(c)* elements that are used by the detection engine which include 3 expression elements that are ultimately used by the detection algorithm. We presented the specifications of 10 agile smells from the *Catalogue of Agile Smells* using the *Agile Smell Schema*.

The *AgileQube App* is a computational system that supports the *Specification*, *Configuration*, *Detection*, and *Validation* phases of the *AgileQube Approach*. The *AgileQube App* is composed of 4 main components:

*(a)* *Specification Module*, a module that aids the *Programmer* in the *Specification* and *Configuration* phases and provides the interfaces to input the agile smells into the application;

*(b)* *ETL Module*, a module that automatically collects data from a project manager system, transforms the extracted data and inputs the translated data into the *Detection Engine*. The current version of the *ETL Module* supports the *ZenHub* platform;

*(c)* *Detection Engine*, a module responsible for performing the *Detection* phase, i.e., detecting the agile smells in a given agile project. This module receives two inputs (*(a)* the data from the assessed agile project represented in the *Agile Project Metamodel* and *(b)* a set of agile smell specifications coded in the *Agile Smell Schema*) and, for each target element and each agile smell specification, it calculates one of the following results: (a) *Preconditions not met*, (b) *Not applicable*, (c) *OK*, or (d) *Not OK*. At the end of this phase, this module produces an *Agile Smells Report (preliminary)*;

*(d)* *Validation Module*, a module that is used in the *Validation* phase and enables a *Team Member* (or someone who knows the project context) to assess the *Agile Smells Report (preliminary)* and decide which of the detected agile smells are false-positive.

To validate the proposed approach and verify whether the research contributions can be used to automatically detect agile smells in an agile project, we conducted 2 case studies. The first case study was a simulated agile project to develop a Manuscrit Submission Management System (MSMS), that we named *Journal Submission System*. The project was hosted on *ZenHub* and had a total of 109 tasks distributed over 7 iterations. We deliberately injected 27 agile smells into the project, configured the *ETL Module*, and requested an agility assessment to the *AgileQube*

*App* that generated an *Agile Smells Report (preliminary)* with all injected agile smells. The approach achieved, in this case study, a precision and recall of 100 percent and the average computation time to generate the *Agile Smells Report* was about 15 seconds.

In the second case study, we selected a real agile project from a medium-size software company that we renamed to *Terminal Operational System*. The project management was hosted on a proprietary platform and had more than 8,000 tasks distributed in about 100 iterations. We exported a subset of tasks from the proprietary platform and manually entered them into the *GitHub* and *ZenHub* projects created to this case study. The selected subset had 279 tasks from 6 iterations. We manually analyzed the selected project structure and found 61 agile smells . The *AgileQube App* was able to load the project data from *ZenHub* and detect 68 agile smells. The approach achieved, in this case study, a precision of 83.8 percent and recall of 93.4 percent. The average computation time to generate the *Agile Smells Report* was about 15 seconds.

The results obtained in the case studies, although not conclusive, indicate the proposed approach was able to automatically detect agile smells in the observed agile projects.

## 9.2 Contributions

The 4 contributions presented in this thesis (*Catalogue of Agile Smells*, *Agile Project Metamodel*, *Agile Smell Schema*, and *AgileQube App*) aid somehow the research in achieving its main goal which was to **propose an agility assessment approach that mitigates the problems P1 to P6 (described in Section 1.2), while providing a solid foundation for defining an infrastructure to support Agility Assessment**. In the remainder of this section, we will analyze the research contributions and discuss how the specific goals presented in Section 1.3 were achieved.

### Goal: *G1. Define a catalogue of agile smells*

The goal G1 (*Define a catalogue of agile smells*) was achieved by the contribution C2 (*Catalogue of Agile Smells*). The catalogue was produced in the *Identification* phase of the *AgileQube Approach* and contains the agile smells that are used in the following phases (*Specification*, *Configuration*, *Detection*, and *Validation*). To identify such agile smells, we conducted a literature review and a survey with practitioners and then we organized the top 10 agile smells as a catalogue. In the catalogue, for each agile smell, we indicated the agile practices and methods that motivated the agile smell and at least one strategy to identify the agile smell in an agile project. This goal mitigates the problem P1 (*Unclear assessment criteria selection*). The relations between P1, G1 and C2 are presented in Figure 9.1.

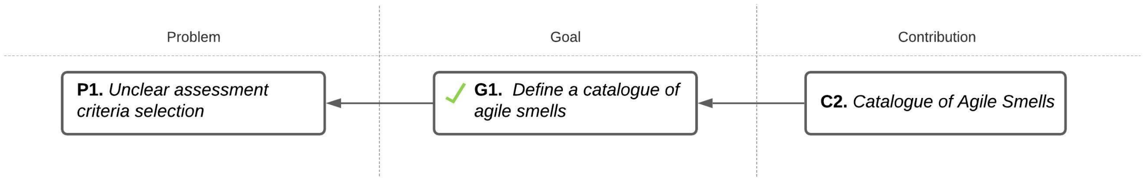

Problem
Goal
Contribution
P1. Unclear assessment criteria selection
✓ G1. Define a catalogue of agile smells
C2. Catalogue of Agile Smells

Figure 9.1: Goal: G1. *Define a catalogue of agile smells*

### Goal: *G2. Define an agile smell representation approach*

Three contributions (C3 *Agile Project Metamodel*, C4 *Agile Smell Schema*, and C5 *AgileQube App/Specification Module*) aid the research achieving the goal G2 (*Define an agile smell representation approach*). The *Agile Project Metamodel* and the *Agile Smell Schema* contains, respectively, the metamodel and the specification language that drive the specification of an agile smell. The *Specification Module* of the *AgileQube App* provides the interfaces necessary to specify an agile smell in a computational system. This goal addresses the problems P2 (*Unclear assessment criteria representation*) and P3 (*Lack of support for adding new assessment criterion*). The relations between P2, P3, G2, C3, C4, and C5 are presented in Figure 9.2.

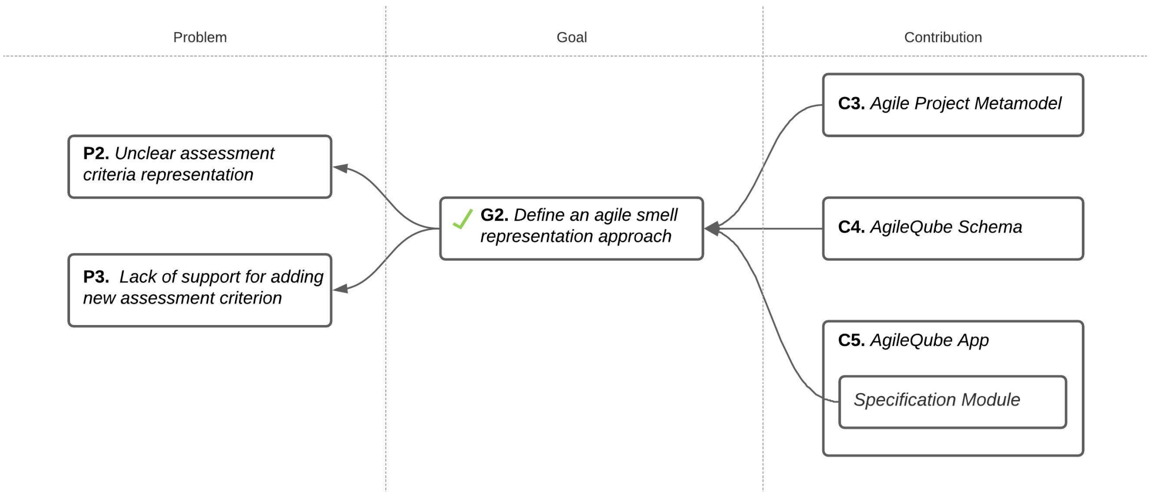

Problem
Goal
Contribution
P2. Unclear assessment criteria representation
P3. Lack of support for adding new assessment criterion
✓ G2. Define an agile smell representation approach
C3. Agile Project Metamodel
C4. AgileQube Schema
C5. AgileQube App
Specification Module

Figure 9.2: Goal: G2. *Define an agile smell representation approach*

**Goal: *G3. Define a data extraction approach***

The goal G3 (*Define a data extraction approach*) is supported by the contributions C3 (*Agile Project Metamodel*) and C5 (*AgileQube App/ETL Module*) in the following way: the *Agile Project Metamodel* is the model supported by the *Detection Engine* while the *ETL Module* is responsible for extracting data from different sources (with different formats) and converting the extracted data into the supported format. The goal G3 mitigates the problems P4 (*Manual data collection and input*), P5 (*Lack of real-time assessment feedback*), and P6 (*Limited Scalability*). The relations between P4, P5, P6, G3, C3, and C5 are presented in Figure 9.3.

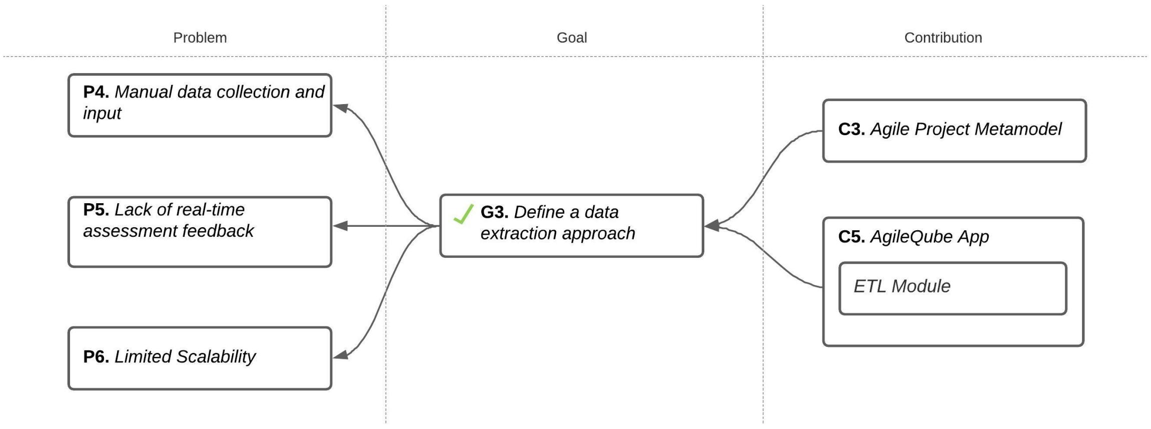

Problem
Goal
Contribution
P4. Manual data collection and input
P5. Lack of real-time assessment feedback
P6. Limited Scalability
G3. Define a data extraction approach
C3. Agile Project Metamodel
C5. AgileQube App
ETL Module

Figure 9.3: Goal: G3. *Define a data extraction approach*

**Goal: *G4. Define and implement a computational system***

The *AgileQube App* and its 4 components (*Specification Module*, *ETL Module*, *Detection Engine*, and *Validation Module*) support the goal G4. This goal solves the problem P6 (*Limited Scalability*). The relations between P6, G4, and C5 are presented in Figure 9.4.

## 9.3 Research Limitations

The limitations of this research were discussed throughout the chapters, specifically when the threats to the validity of the contributions were presented. It is worth mentioning the following limitations:

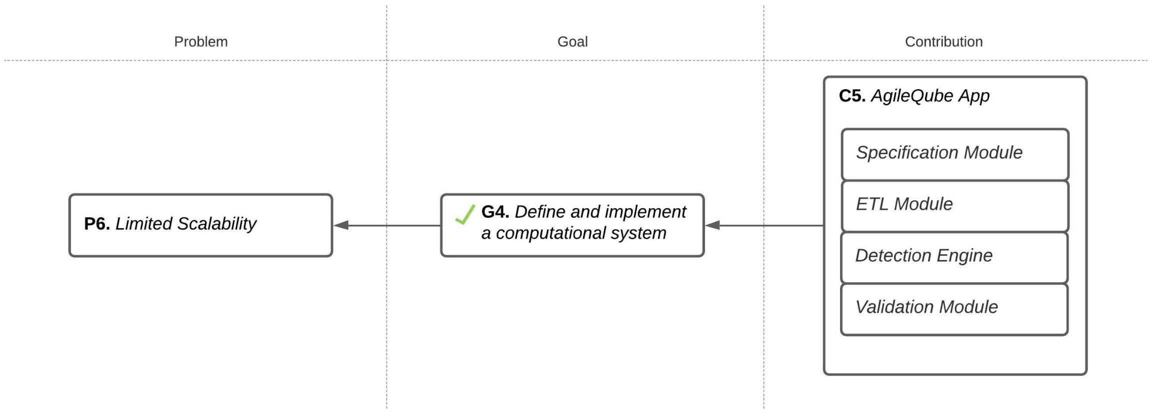


Figure 9.4: Goal: G4. *Define and implement a computational system*

### *Catalogue of Agile Smells*

- When we conducted the literature review to identify agile smells, there was no use of the term "smell" in the literature. We mitigated this issue by defining objective criteria to identify in the selected papers situations that may impair the adoption of agile practices.
- A significant part of the body of knowledge about Agile Development is created by software engineering practitioners that usually do not publish in academic forums [179]. Hence, we decided to include grey literature sources (non-peer-reviewed material) in literature review to identify agile smells.
- The survey sampling size is not representative enough to allow us to affirm that the set of identified agile smells represents the most relevant ones. So, there may be some variation in the ranked list if we conduct a survey with a more representative sampling.

### *Agile Project Metamodel*

- Although the *Agile Project Metamodel* can be used in a wide range of research related to Agile Development, the motivation to identify such a model came from the need to represent the data from an agile project for the purposes of agility assessment. We are not claiming the metamodel can represent all aspects from an agile project and even the possible inclusion of new agile smells into the *AgileQube Approach* may reveal the need for new concepts in the metamodel.

***AgileQube App***

- Although the *ETL Module* was designed to enable the approach to collect data from a wide range of project management tools such as *ZenHub*, HuBoard, Waffle.io, Redmine, BaseCamp, and Jira, the implementation presented in this study supports only the *ZenHub* platform.

**Case Studies**

- The difficulty of finding open access projects hosted on *GitHub* or *ZenHub* that could be used as case studies to validate whether the proposed approach can automatically detect agile smells;

- The difficulty of conducting experimental studies to assess the usability and applicability of proposed approach.

## 9.4 Future Work

The future work intended to be done includes the following:

1. **Technical debts** - Investigate the potential technical debts caused by an agile smell. This includes identifying the potential technical debt, understanding how to measure it, and including it in the catalogue and in the schema.

2. **New Agile Smells** - continue the evolution of the *Catalogue of Agile Smells* by identifying and cataloguing new agile smells.

3. **Survey with a representative sampling** - Conduct a survey with a more representative sampling in order to confirm (or adjust) the rank of the most relevant agile smells presented in this research. Such a survey mitigates the conclusion validity threat present in this research.

4. **New Empirical Studies** – Conduct empirical studies to assess the aplicability and usability of the proposed approach.

5. **Agile Smells and Agility Maturity Models** - Investigate the potential relationship between the agile smells and an agility maturity model.

6. **Evolve the ETL Module** - Evolve the *ETL Module* to support other project management platforms besides *ZenHub*.

# Bibliography

[1] PEFFERS, K., TUUNANEN, T., ROTHENBERGER, M., et al. "A design science research methodology for information systems research", *J. Manage. Inf. Syst.*, v. 24, n. 3, pp. 45–77, dez. 2007. ISSN: 0742-1222. doi: 10.2753/MIS0742-1222240302. Available at: <`http://dx.doi.org/10.2753/MIS0742-1222240302`>.

[2] OMG. *Business process model and notation (BPMN).* Final specification, OMG, jan 2011. http://www.omg.org/spec/BPMN/2.0/.

[3] RAO, K. N., NAIDU, G. K., CHAKKA, P. "A study of the agile software development methods, applicability and implications in industry", *International Journal of Software Engineering and its applications*, v. 5, n. 2, pp. 35–45, 2011.

[4] MEYER, B. *Agile!: the good, the hype and the ugly*, v. 9783319051550, *Agile!: the good, the hype and the ugly.* Cham, Spring, 2014.

[5] TRIPP, J., ARMSTRONG, D. "Agile methodologies: organizational adoption motives, tailoring, and performance", *Journal of Computer Information Systems*, v. 58, pp. 1–10, 10 2016. doi: 10.1080/08874417.2016.1220240.

[6] TARWANI, S., CHUG, A. "Agile methodologies in software maintenance: A systematic review", *Informatica*, v. 40, n. 4, 2016.

[7] VERSIONONE. "The 4th annual state of agile report$^{TM}$ 2009". https://stateofagile.com/, 2009. Accessed: 2020-09-25.

[8] VERSIONONE. "The 14th annual state of agile report$^{TM}$ 2020". https://stateofagile.com/, 2020. Accessed: 2020-09-25.

[9] SIDKY, A., ARTHUR, J., BOHNER, S. "A disciplined approach to adopting agile practices: the agile adoption framework", *Innovations in systems and software engineering*, v. 3, n. 3, pp. 203–216, 2007.

[10] QUMER, A., HENDERSON-SELLERS, B. "A framework to support the evaluation, adoption and improvement of agile methods in practice", *Journal of Systems and Software*, v. 81, n. 11, pp. 1899–1919, 2008.

[11] ELSSAMADISY, A. *Agile Adoption Patterns: A Roadmap to Organizational Success (Adobe ebook)*. Addison-Wesley Professional, 2008.

[12] ROHUNEN, A., RODRIGUEZ, P., KUVAJA, P., et al. "Approaches to agile adoption in large settings: a comparison of the results from a literature analysis and an industrial inventory". In: *International Conference on Product Focused Software Process Improvement*, pp. 77–91. Springer, 2010.

[13] HAJJDIAB, H., TALEB, A. "Adopting Agile Software Development: Issues and Challenges", *International Journal of Managing Value and Supply Chains*, v. 2, pp. 1–10, 09 2011. doi: 10.5121/ijmvsc.2011.2301.

[14] BARLOW, J. B., GIBONEY, J., KEITH, M. J., et al. "Overview and guidance on agile development in large organizations", *Communications of the Association for Information Systems*, v. 29, n. 2, pp. 25–44, 2011.

[15] GANDOMANI, T. J., NAFCHI, M. Z. "An empirically-developed framework for Agile transition and adoption: A Grounded Theory approach", *Journal of Systems and Software*, v. 107, pp. 204–219, 2015.

[16] DE SOUZA BERMEJO, P., ZAMBALDE, A., TONELLI, A., et al. "Agile Principles and Achievement of Success in Software Development: A Quantitative Study in Brazilian Organizations", *Procedia Technology*, v. 16, pp. 718–727, 12 2014. doi: 10.1016/j.protcy.2014.10.021.

[17] ELORANTA, V.-P., KOSKIMIES, K., MIKKONEN, T. "Exploring ScrumBut – An Empirical Study of Scrum Anti-Patterns", *Information and Software Technology*, v. 74, 12 2015. doi: 10.1016/j.infsof.2015.12.003.

[18] LÓPEZ-MARTÍNEZ, J., JUÁREZ-RAMÍREZ, R., HUERTAS, C., et al. "Problems in the adoption of agile-scrum methodologies: A systematic literature review". In: *2016 4th international conference in software engineering research and innovation (conisoft)*, pp. 141–148. IEEE, 2016.

[19] GREGORY, P., BARROCA, L., SHARP, H., et al. "The challenges that challenge: Engaging with agile practitioners' concerns", *Information and Software Technology*, v. 77, pp. 92 – 104, 2016. ISSN: 0950-5849. doi: https://doi.org/10.1016/j.infsof.2016.04.

006. Available at: <http://www.sciencedirect.com/science/article/pii/S0950584916300623>.

[20] AMBLER, S. "IT project success rates survey results". http://www.ambysoft.com/surveys/success2018.html, 2018. Accessed: 2019-10-01.

[21] ALLIANCE, S. "Learn about scrum". https://www.scrumalliance.-org/why-scrum, 2016. Accessed: 2017-12-01.

[22] OZCAN-TOP, O., DEMIRÖRS, O. "A Reference Model for Software Agility Assessment: AgilityMod". In: Rout, T., O'Connor, R. V., Dorling, A. (Eds.), *Software Process Improvement and Capability Determination*, pp. 145–158, Cham, 2015. Springer International Publishing. ISBN: 978-3-319-19860-6.

[23] COCKBURN, A. *Agile software development.* Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN: 0-201-69969-9.

[24] HIGHSMITH, III, J. A. *Adaptive software development: a collaborative approach to managing complex systems.* New York, NY, USA, Dorset House Publishing Co., Inc., 2000. ISBN: 0-932633-40-4.

[25] PACKLICK, J. "The agile maturity map a goal oriented approach to agile improvement". In: *Agile 2007 (AGILE 2007)*, pp. 266–271. IEEE, 2007.

[26] ADALI, O. E., ÖZCAN-TOP, Ö., DEMIRÖRS, O. "Evaluation of Agility Assessment Tools: A Multiple Case Study". In: Clarke, P. M., O'Connor, R. V., Rout, T., et al. (Eds.), *Software Process Improvement and Capability Determination*, pp. 135–149, Cham, 2016. Springer International Publishing. ISBN: 978-3-319-38980-6.

[27] SOUNDARARAJAN, S., ARTHUR, J. D. "A structured framework for assessing the "goodness" of agile methods". In: *2011 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*, pp. 14–23. IEEE, 2011.

[28] STORM-CONSULTING. *Agile Enterprise Survey*, 2008. Available at: <http://www.storm-consulting.com/agile-enterprise-survey/>.

[29] KREBS, B. *Agile Journey Index*, 2011. Available at: <http://www.agiledimensions.com/blog/>.

[30] LAGESTEE, L. *Agile Health Dashboard*, 2012. Available at: <http://illustratedagile.com/2012/09/25/how-to-measure-team-agility/>.

[31] INFOTECH. *Agile Process Assessment Tool*, 2013. Available at: <https://www.infotech.com/research/it-agile-process-assessment-tool>.

[32] COHN, M., RUBIN, K. *Comparative Agile*, 2015. Available at: <https://www.comparativeagility.com/capabilities/agile-assessment>.

[33] TOUSIGNANT, D. *Agile Maturity Assessment*, 2019. Available at: <https://capeprojectmanagement.com/individual-assessment/>.

[34] TURETKEN, O., ELGAMMAL, A., VAN DEN HEUVEL, W., et al. "Capturing Compliance Requirements: A Pattern-Based Approach", *IEEE Software*, v. 29, n. 3, pp. 28–36, 2012.

[35] LY, L. T., MAGGI, F. M., MONTALI, M., et al. "Compliance monitoring in business processes: functionalities, application, and tool-support", *Information Systems*, v. 54, pp. 209 – 234, 2015. ISSN: 0306-4379. doi: https://doi.org/10.1016/j.is.2015.02.007. Available at: <http://www.sciencedirect.com/science/article/pii/S0306437915000459>.

[36] ADALI, O. E., ÖZCAN-TOP, Ö., DEMIRÖRS, O. "Evaluation of agility assessment tools: a multiple case study". In: *International Conference on Software Process Improvement and Capability Determination*, pp. 135–149. Springer, 2016.

[37] MCCALLA, M., GIFFORD, J. *Lean Agile Intelligence*, 2016. Available at: <https://www.leanagileintelligence.com/>.

[38] MOHA, N., GUEHENEUC, Y., LEDUC, P. "Automatic Generation of Detection Algorithms for Design Defects". In: *21st IEEE/ACM International Conference on Automated Software Engineering (ASE'06)*, pp. 297–300, 2006.

[39] AGILE, S. *Team and Technical Agility Self-Assessment*, 2012. Available at: <https://www.scaledagileframework.com/metrics/#T4>.

[40] ELIASSEN-GROUP. *Enterprise Agility Maturity Matrix*, 2013. Available at: <`http://blog.eliassen.com/introducing-the-enterprise-agility-maturity-matrix`>.

[41] ULLAH, K. W., AHMED, A. S., YLITALO, J. "Towards Building an Automated Security Compliance Tool for the Cloud". In: *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 1587–1593, 2013.

[42] DIMYADI, J., AMOR, R. "Automating Conventional Compliance Audit Processes". In: *Product Lifecycle Management and the Industry of the Future*, pp. 324–334, Cham, 2017. Springer International Publishing.

[43] COLLIER, B., DEMARCO, T., FEAREY, P. "A defined process for project post mortem review", *IEEE Software*, v. 13, n. 4, pp. 65–72, 1996. ISSN: 0740-7459.

[44] PARK, J. S., SPETKA, E., RASHEED, H., et al. "Near-Real-Time Cloud Auditing for Rapid Response". In: *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, pp. 1252–1257, 2012.

[45] FOWLER, M., BECK, K., BRANT, J., et al. *Refactoring: improving the design of existing code.* Addison-Wesley Professional, 1999.

[46] TELEMACO, U., OLIVEIRA, T. "BPMN-R: An extension to BPMN to Represent Software Process Rules: doctoral symposium". In: *XV Workshop de Teses e Dissertações em Qualidade de Software (WTDQS 2017)*, 2017.

[47] TELEMACO, U., OLIVEIRA, T., ALENCAR, P., et al. "A catalog of bad agile smells for agility assessment". In: *Proceedings of the 2019 Ibero-American Conference on Software Engineering*, CIbSE 2019, pp. 30–43, 2019.

[48] TELEMACO, U., OLIVEIRA, T., ALENCAR, P., et al. "A metamodel for representing agile software development projects". In: *Proceedings of the 2019 Ibero-American Conference on Software Engineering*, CIbSE 2019, 2019.

[49] TELEMACO, U., OLIVEIRA, T., ALENCAR, P., et al. "A Catalogue of Agile Smells for Agility Assessment", *IEEE Access*, v. 8, pp. 79239–79259, 2020.

[50] LU, R., SADIQ, S., GOVERNATORI, G. "Compliance aware business process design". In: *Business Process Management Workshops*, Springer Nature, pp. 120–131, 2008.

[51] DE MELLO, R. M., MOTTA, R. C., TRAVASSOS, G. H. "A Checklist-Based Inspection Technique for Business Process Models". In: *International Conference on Business Process Management*, pp. 108–123. Springer, 2016.

[52] RUNESON, P., HÖST, M. "Guidelines for conducting and reporting case study research in software engineering", *Empirical software engineering*, v. 14, n. 2, pp. 131, 2009.

[53] BECK, K., BEEDLE, M., VAN BENNEKUM, A., et al. "Manifesto for agile software development". http://www.agilemanifesto.org/, 2001.

[54] LARMAN, C., BASILI, V. R. "Iterative and incremental developments. a brief history", *Computer*, v. 36, n. 6, pp. 47–56, 2003.

[55] BECK, K. "Embracing change with extreme programming", *Computer*, v. 32, n. 10, pp. 70–77, 1999.

[56] BECK, K. *Extreme programming explained: embrace change.* Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 2000. ISBN: 0-201-61641-6.

[57] CUNNINGHAM, W. "Extreme programming". http://www.-extremeprogramming.org/, 1999. Accessed: 2017-12-01.

[58] SCHWABER, K. "SCRUM development process". In: *Business Object Design and Implementation*, Springer London, pp. 117–134, 1997.

[59] SCHWABER, K., BEEDLE, M. *Agile software development with scrum.* 1st ed. Upper Saddle River, NJ, USA, Prentice Hall PTR, 2001. ISBN: 0130676349.

[60] BERTEIG, M. "Rules of scrum". http://www.agileadvice.com/rules-of-scrum/, 2015. Accessed: 2017-12-01.

[61] PALMER, S. R., FELSING, M. *A practical guide to feature-driven development.* Pearson Education, 2001. ISBN: 0130676152.

[62] LUCA, J. D. "Feature driven development FDD". http://www.-featuredrivendevelopment.com/, 1999. Accessed: 2017-12-01.

[63] STAPLETON, J. *Dynamic systems development method: the method in practice.* Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 1997. ISBN: 0201178893.

[64] ABRAHAMSSON, P., SALO, O., RONKAINEN, J., et al. *Agile software development methods - review and analysis.* Relatório Técnico 478, VTT Publications, Espoo, Finland, 2002.

[65] TAKEUCHI, H., NONAKA, I. *The New New Product Development Game*, 1986.

[66] 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.

[67] FRASER, S., BOEHM, B., JÄRKVIK, J., et al. "How Do Agile/XP Development Methods Affect Companies?" In: Abrahamsson, P., Marchesi, M., Succi, G. (Eds.), *Extreme Programming and Agile Processes in Software Engineering*, pp. 225–228, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN: 978-3-540-35095-8.

[68] GANDOMANI, T. J., ZULZALIL, H., GHANI, A. A. A., et al. "Obstacles in moving to agile software development methods at a glance", *Journal of Computer Science*, v. 9, n. 5, pp. 620, 2013.

[69] GANDOMANI, T. J., ZULZALIL, H., GHANI, A., et al. "Important considerations for agile software development methods governance." *Journal of Theoretical & Applied Information Technology*, v. 55, n. 3, pp. 345–351, 2013.

[70] SAHOTA, M. *An Agile Adoption and Transformation Survival Guide.* Lulu. com, 2012.

[71] AMBLER, S. W., LINES, M. *Disciplined agile delivery: a practitioner's guide to agile software delivery in the enterprise.* IBM press, 2012.

[72] YATZECK, E. "A corporate agile 10-point checklist". http://pagilista.blogspot.com/2012/12/a-corporate-agile-10-point-checklist.html, Dec 2012. Accessed: 2019-06-30.

[73] WILLIAMS, L., RUBIN, K., COHN, M. "Driving process improvement via comparative agility assessment". In: *2010 Agile Conference*, pp. 3–10. IEEE, 2010.

[74] SFIRLOGEA, S., GEORGESCU, F. *Retropoly*, 2017. Available at: <`https://www.agilepractice.eu/retropoly/`>.

[75] LINDERS, B. *The Agile Self-assessment Game.* Leanpub, 2019.

[76] JANLÉN, J. *Team Barometer*, 2014. Available at: <https://blog.crisp.se/2014/01/30/jimmyjanlen/team-barometer-self-evaluation-tool>.

[77] BRITSCH, M. *Agility Questionnaire*, 2017. Available at: <https://thedigitalbusinessanalyst.co.uk/agility-questionnaire-130b03133b98>.

[78] ACHOUIANTZ, C. *Depth of Kanban*, 2013. Available at: <http://leanagileprojects.blogspot.com/2013/03/depth-of-kanban-good-coaching-tool.html>.

[79] YERET, Y. *Lean/Agile Depth Assessment Checklist A3*, 2013. Available at: <https://www.slideshare.net/yyeret/leanagile-depth-assessment>.

[80] ALBRECHT, K., EDDINGS, S. *Measure.team*, 2020. Available at: <https://measure.team/>.

[81] KNIBERG, H. "Scrum checklist". http://www.crisp.se/scrum/checklist, 2012. Accessed: 2019-06-30.

[82] WATERS, K. *How agile are you? ((Take This 42 Point Test))*, 2008. Available at: <https://www.101ways.com/2008/01/21/how-agile-are-you-take-this-42-point-test/>.

[83] HERMIDA, S. "A better team". http://www.abetterteam.org/, 2009. Accessed: 2019-06-30.

[84] BONAMASSA, M. *Agile Adoption Interview*, 2018. Available at: <https://pladcloud.typeform.com/to/HjcVKG>.

[85] HOFFMANN, R., BONA, T., PETIT, S. *Agile Alert*, 2018. Available at: <https://huz.de/en/agile-alert/>.

[86] NOWINSKI, P. *Agile Assessment*, 2016. Available at: <http://piotr-nowinski.pl/agile-assessment/>.

[87] LEWIS, R., WENDLER, R. *Agile Enterprise Survey*, 2016. Available at: <http://www.storm-consulting.com/agile-enterprise-survey/>.

[88] SFIRLOGEA, S., GEORGESCU, F. *Agile Excellerate*, 2020. Available at: <https://www.agilepractice.eu/agile-excellerate/>.

[89] BPMI, B. I. *Agile Skills Self-Assessment*, 2019. Available at: <https://www.bpminstitute.org/skills-self-assessments#>.

[90] GUNNERSON, E. *Agile Team Evaluation*, 2015. Available at: <https://docs.microsoft.com/en-us/archive/blogs/ericgu/agile-team-evaluation>.

[91] CAMPBELL, B., MACIVER, R. "Agility maturity self assessment". http://www.robbiemaciver.com/documents/presentations/A2010-Agile%20Maturity%20Self-Assessment.pdf, 2010. Accessed: 2019-06-30.

[92] RIBEIRO, E. "Agility maturity self assessment survey". https://beyondleanagile.com/2015/12/08/agile-maturity-self-assessment-survey-published-at-scrumalliance/, 2015. Accessed: 2019-06-30.

[93] HENDRICKSON, E. *Back-of-a-Napkin Agile Assessment*, 2008. Available at: <https://testobsessed.com/2008/11/back-of-a-napkin-agile-assessment/>.

[94] BALBES, M. *How Agile Are You? Let's Actually Measure It!*, 2015. Available at: <https://adtmag.com/articles/2015/12/15/balbes-agile-model-0-intro.aspx>.

[95] SCHUMACHER, D. *Borland Agile Assessment 2009*, 2009. Available at: <https://borland.typepad.com/agile_transformation/2009/03/borland-agile-assessment-2009.html>.

[96] BURLTON, R. T., ROSS, R. G., ZACHMAN, J. A. *Business Agility Manifesto*, 2018. Available at: <https://busagilitymanifesto.org/accompaniments/supplements/diagnostics>.

[97] WOLPERS, S. *Cargo Cult Agile Checklist*, 2016. Available at: <https://age-of-product.com/cargo-cult-agile-state-agile-checklist-organization/>.

[98] ERANDE, A. S., VERMA, A. K. "Measuring agility of organizations-a comprehensive agility measurement tool (CAMT)", *International journal of applied management and technology*, v. 6, n. 3, 2008.

[99] OF DEFENSE DOD, D. *DIB Guide: Detecting Agile BS*, 2018. Available at: <https://media.defense.gov/2018/Oct/09/2002049591/-1/-1/0/DIB_DETECTING_AGILE_BS_2018.10.05.PDF>.

[100] RIBEIRO, E. *Enterprise Business Agility Maturity Survey*, 2018. Available at: <https://beyondleanagile.com/2018/12/12/enterprise-business-agility-maturity-survey/>.

[101] NIELSEN, J. *Five key numbers to gauge your agile engineering efforts*, 2011. Available at: <https://www.slideshare.net/jeffreymads/five-key-numbers-to-gauge-your-agile-engineering-efforts-798034

[102] GAO, U. G. A. O. *Agile Assessment Guide*, 2020. Available at: <https://www.gao.gov/assets/710/709711.pdf>.

[103] FINITE. *How Agile are you? A 50 Point Test*, 2019. Available at: <https://www.finite.com.au/blog/2019/08/how-agile-are-you/>.

[104] IBM. "IBM devops self-assessment". https://devopsassessment.mybluemix.net//, 2008. Accessed: 2019-06-30.

[105] LITTLE, J. H. *Joe's Unofficial Scrum Checklist*, 2012. Available at: <http://agileconsortium.pbworks.com/w/file/fetch/66642311/Joe\%E2\%80\%99s\%20Unofficial\%20Scrum\%20CheckList\%20V13.pdf>.

[106] SEUFFERT, M. *Karlskrona test online*, 2019. Available at: <https://mayberg.se/karlskrona-test-online>.

[107] CHIVA, G. *Kanban Maturity Assessment*, 2019. Available at: <https://aktiasolutions.com/kanban-maturity-assessment/>.

[108] LEBOW, J. *Agile Assessment: Test Your Team's Agility*, 2018. Available at: <https://digital.ai/resources/agile-101/agile-assessment>.

[109] VODDE, B., SUTHERLAND, J. *Nokia Test*, 2010. Available at: <https://www.scruminc.com/official-scrumbutt-test-otherwise-known/>.

[110] SOUNDARARAJAN, S. *Assessing agile methods: investigating adequacy, capability, and effectiveness (an objectives, principles, strategies approach)*. Tese de Doutorado, Virginia Tech, 2013.

[111] SCRUM.ORG. *Open Assessments*, 2020. Available at: <https://www.scrum.org/open-assessments>.

[112] AGILE, S. *Organizational Agility Self-Assessment*, 2012. Available at: <https://www.scaledagileframework.com/metrics/#PF9>.

[113] SHOUKATH, N. *Are you really agile?*, 2012. Available at: <https://blog.people10.com/are-you-really-agile-a-free-assessment-here/>.

[114] SO, C., SCHOLL, W. "Perceptive Agile Measurement: New Instruments for Quantitative Studies in the Pursuit of the Social-Psychological Effect of Agile Practices". In: Abrahamsson, P., Marchesi, M., Maurer, F. (Eds.), *Agile Processes in Software Engineering and Extreme Programming*, pp. 83–93, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN: 978-3-642-01853-4.

[115] PARRY, C. *Quick self-assessment of your organization's agility*, 2009. Available at: <http://www.signetconsulting.com/action_items/assessment.php>.

[116] HAWKS, D. *Scrum Assessment Series*, 2013.

[117] JAMES, M. *ScrumMaster Checklist*, 2007. Available at: <https://scrummasterchecklist.org/pdf/ScrumMaster_Checklist_12_unbranded.pdf>.

[118] ROTHMAN, J. *Self assessment tool for transitioning to agile*, 2013. Available at: <https://www.jrothman.com/mpd/agile/2013/04/self-assessment-tool-for-transitioning-to-agile/>. Accessed: 2019-06-30.

[119] KNIBERG, H., LINDWALL, K. *Squad Health Check Model*, 2014. Available at: <https://engineering.atspotify.com/2014/09/16/squad-health-check-model/>.

[120] VERWIJS, C. *TeamMetrics*, 2017. Available at: <https://teammetrics.theliberators.com/>.

[121] SCHOOTS, H., SCHUURKES, J. *Test Maturity Card Game*, 2017. Available at: <https://www.huibschoots.nl/wordpress/wp-content/uploads/2017/02/Test-Improvement-Huib-Schoots-Joep-Schuurkes.pdf>.

[122] SHORE, J., WARDEN, S. *The art of agile development: pragmatic guide to agile software development.* " O'Reilly Media, Inc.", 2007.

[123] SPOLSKY, J. *The Joel Test: 12 Steps to Better Code*, 2000. Available at: <https://www.joelonsoftware.com/2000/08/09/the-joel-test-12-steps-to-better-code/>.

[124] HOGAN, B. *Visual Management Self-Assessment*, 2017. Available at: <http://agileben.com/blog/kanban-online-assessment>.

[125] YODIZ. *Team Agility Self Assessment*, 2017. Available at: <https://www.yodiz.com/blog/how-agile-is-your-team-take-our-team-agility-self-assessment-to

[126] NAWROCKI, J., WALTER, B., WOJCIECHOWSKI, A. "Toward maturity model for extreme programming". In: *Proceedings 27th EUROMICRO Conference. 2001: A Net Odyssey*, pp. 233–239. IEEE, 2001.

[127] TEAM, C. P. "CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Version 1.02", *CMU/SEI*, 2000.

[128] LUI, K. M., CHAN, K. C. "A Road Map for Implementing eXtreme Programming". In: *Software Process Workshop*, pp. 474–481. Springer, 2005.

[129] PATEL, C., RAMACHANDRAN, M. "Agile maturity model (AMM): a software process improvement framework for agile software development practices", *International Journal of Software Engineering, IJSE*, v. 2, n. 1, pp. 3–28, 2009.

[130] ISO/IEC 15504. *ISO/IEC 15504: Information Technology - Process Assessment, Part 1 to Part 5.* Standard, International Organization for Standardization, Geneva, CH, 2002.

[131] TURETKEN, O., STOJANOV, I., TRIENEKENS, J. J. "Assessing the adoption level of scaled agile development: a maturity model for Scaled Agile Framework", *Journal of Software: Evolution and process*, v. 29, n. 6, pp. e1796, 2017.

[132] LEFFINGWELL, D. *SAFe 4.0 Reference Guide: Scaled Agile Framework for Lean Software and Systems Engineering.* Addison-Wesley Professional, 2016. ISBN: 0134510542.

[133] PETTIT, R. *Innovation Maturity Model*, 2006. Available at: <http://www.rosspettit.com/2007/02/innovation-maturity-model.html>.

[134] AMBLER, S. W. "The agile scaling model (ASM): adapting agile methods for complex environments", *Environments*, pp. 1–35, 2009.

[135] QUMER, A., HENDERSON-SELLERS, B. "Measuring agility and adaptibility of agile methods: A 4 dimensional analytical tool". In: *The IADIS international conference on applied computing 2006*. IADIS Press, 2006.

[136] SURESHCHANDRA, K., SHRINIVASAVADHANI, J. "Adopting agile in distributed development". In: *2008 IEEE International Conference on Global Software Engineering*, pp. 217–221. IEEE, 2008.

[137] ESFAHANI, H. C. *Transitioning to agile: A framework for pre-adoption analysis using empirical knowledge and strategic modeling*. Tese de Doutorado, 2012.

[138] GLOVER, B. *Scrum Checklist 2012*, 2012. Available at: <`https://www.infoq.com/minibooks/scrum-checklists/`>.

[139] MOHA, N., GUEHENEUC, Y.-G., DUCHIEN, L., et al. "Decor: a method for the specification and detection of code and design smells", *IEEE Transactions on Software Engineering*, v. 36, n. 1, pp. 20–36, 2009.

[140] FOUNDATION, E. "OpenUp methodology". http://epf.eclipse.org/wikis/openup/, 2012. Accessed: 2017-12-01.

[141] SPÍNOLA, R. O., DIAS-NETO, A. C., TRAVASSOS, G. H. "Abordagem para desenvolver tecnologia de software com apoio de estudos secundários e primários". In: *Experimental Software Engineering Latin American Workshop (ESELAW)*, 2008.

[142] KITCHENHAM, B., CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Relatório Técnico EBSE 2007-001, Keele University and Durham University Joint Report, Jul 2007.

[143] PAI, M., MCCULLOCH, M., GORMAN, J., et al. "Systematic reviews and meta-analyses: an illustrated, step-by-step guide." *The Medical Journal of India*, 2004.

[144] 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, Washington, DC, USA, 2001. IEEE Computer Society.

[145] LINDVALL, M., BASILI, V. R., BOEHM, B. W., et al. "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*, London, UK, UK, 2002. Springer-Verlag.

[146] TSOURVELOUDIS, N. C., VALAVANIS, K. P. "On the measurement of enterprise agility", *Journal of Intelligent and Robotic Systems*, v. 33, n. 3, pp. 329–342, 2002.

[147] GILL, A., HENDERSON-SELLERS, B. "Measuring agility and adaptibility of agile methods: a 4 dimensional analytical tool". In: *The IADIS international conference on applied computing 2006*. IADIS Press, 2006.

[148] LI, J., BURNHAM, J. F., LEMLEY, T., et al. "Citation analysis: comparison of Web of Science®, Scopus®, SciFinder®, and Google Scholar", *Journal of electronic resources in medical libraries*, v. 7, n. 3, pp. 196–217, 2010.

[149] MAURER, F., MARTEL, S. "Extreme programming: rapid development for web-based applications", *IEEE Internet Computing*, v. 6, n. 1, pp. 86–90, 2002.

[150] 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.

[151] MARTIN, R. C. *Agile software development: principles, patterns, and practices*. Upper Saddle River, NJ, USA, Prentice Hall PTR, 2003. ISBN: 0135974445.

[152] WILLIAMS, L. "Agile software development methodologies and practices". In: *Advances in Computers*, v. 80, Elsevier, pp. 1–44, 2010.

[153] ALZOABI, Z. "Agile software: body of knowledge". In: *Business Intelligence and Agile Methodologies for Knowledge-Based Organizations: Cross-Disciplinary Applications*, IGI Global, pp. 14–34, 2012.

[154] MORAN, A. "Agile software development". In: *Agile Risk Management*, Springer International Publishing, pp. 1–16, Cham, 2014.

[155] 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.

[156] GHANI, I., BELLO, M. “Agile adoption in IT organizations”, *KSII Transactions on Internet and Information Systems*, v. 9, n. 8, pp. 3231–3248, 2015.

[157] 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.

[158] 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*, v. 7, n. 3, pp. 167–180, 2017.

[159] SLETHOLT, M., HANNAY, J., PFAHL, D., et al. “What do we know about scientific software development’s agile practices?” *Computing in Science and Engineering*, v. 14, n. 2, pp. 24–36, 2012.

[160] PAPATHEOCHAROUS, E., ANDREOU, A. “Empirical evidence and state of practice of software agile teams”, *Journal of Software: Evolution and Process*, v. 26, n. 9, pp. 855–866, 2014.

[161] UIKEY, N., SUMAN, U. “Tailoring for agile methodologies: a framework for sustaining quality and productivity”, *International Journal of Business Information Systems*, v. 23, n. 4, pp. 432–455, 2016.

[162] KROPP, M., MEIER, A., BIDDLE, R. “Agile Practices, Collaboration and Experience”. In: Abrahamsson, P., Jedlitschka, A., Nguyen Duc, A., et al. (Eds.), *Product-Focused Software Process Improvement*, pp. 416–431, Cham, 2016. Springer International Publishing. ISBN: 978-3-319-49094-6.

[163] JAIN, R., SUMAN, U. “Effectiveness of agile practices in global software development”, *International Journal of Grid and Distributed Computing*, v. 9, n. 10, pp. 231–248, 2016.

[164] 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*, v. 2018-June, pp. 1–6, 2018.

[165] VALLON, R., DA SILVA ESTÁCIO, B., PRIKLADNICKI, R., et al. “Systematic literature review on agile practices in global software development”, *Information and Software Technology*, v. 96, pp. 161–180, 2018.

[166] NISAR, M., HAMEED, T. “Agile methods handling offshore software development issues”. In: *Proceedings of INMIC 2004 - 8th International Multitopic Conference*, pp. 417–422, 2004. doi: 10.1109/INMIC.2004.1492915.

[167] CHAGAS, L. F., DE CARVALHO, D. D., LIMA, ADAILTON MAGALHÃES AN REIS, C. A. L. “Systematic Literature Review on the Characteristics of Agile Project Management in the Context of Maturity Models”. In: Mitasiunas, A., Rout, T., O’Connor, R. V., et al. (Eds.), *Software Process Improvement and Capability Determination*, pp. 177–189, Cham, 2014. Springer International Publishing. ISBN: 978-3-319-13036-1.

[168] CAO, L., RAMESH, B. “Agile software development: ad hoc practices or sound principles?” *IT professional*, v. 9, n. 2, pp. 41–47, 2007.

[169] 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.

[170] MISRA, S. C., KUMAR, V., KUMAR, U. “Identifying some important success factors in adopting agile software development practices”, *J. Syst. Softw.*, v. 82, n. 11, pp. 1869–1890, nov 2009.

[171] LI, J. “Research and practice of agile unified process”. In: *ICSTE 2010 - 2010 2nd International Conference on Software Technology and Engineering, Proceedings*, v. 2, pp. V2340–V2343, 2010.

[172] MCMAHON, P. “Extending agile methods: a distributed project and organizational improvement perspective”, *CrossTalk*, , n. 5, pp. 16–19, 2005.

[173] SHI, Z., CHEN, L., CHEN, T.-E. “Agile planning and development methods”. In: *ICCRD2011 - 2011 3rd International Conference on Computer Research and Development*, v. 1, pp. 488–491, 2011.

[174] KAJORNBOON, A. B. “Using interviews as research instruments”, *E-journal for Research Teachers*, v. 2, n. 1, pp. 1–9, 2005.

[175] OISHI, S. *How to conduct in-person interviews for surveys*. SAGE Publications Inc., 2003.

[176] GHAZI, A. N., PETERSEN, K., REDDY, S. S. V. R., et al. “Survey research in software engineering: problems and strategies”, *arXiv preprint arXiv:1704.01090*, 2017.

[177] GAMMA, E., HELM, R., JOHNSON, R., et al. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[178] COHEN, P., COHEN, J., AIKEN, L. S., et al. “The problem of units and the circumstance for POMP”, *Multivariate behavioral research*, v. 34, n. 3, pp. 315–346, 1999.

[179] GLASS, R. L., DEMARCO, T. *Software Creativity 2.0*. Atlanta, Georgia, Developer Dot Star Books, 2006. ISBN: 0977213315.

[180] SANTOS, R. M. D. S. *Mapeamento entre Representações de Processo e Projeto de Software*. Tese de Doutorado, UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2019.

[181] TRAVASSOS, G. H., D. SANTOS, P. S. M., MIAN, P. G., et al. “An environment to support large scale experimentation in software engineering”. In: *13th IEEE International Conference on Engineering of Complex Computer Systems (iceccs 2008)*, pp. 193–202, March 2008. doi: 10.1109/ICECCS.2008.30.

[182] STEENWEG, R., KUHRMANN, M., MÉNDEZ FERNÁNDEZ, D. “Software engineering process metamodels–a literature review”, *Technische Universität München, Tech. Rep. TUM-I1220*, 2012.

[183] BENDRAOU, R., JEZEQUEL, J.-M., GERVAIS, M.-P., et al. “A comparison of six UML-based languages for software process modeling”, *IEEE Transactions on Software Engineering*, v. 36, n. 5, pp. 662–675, sep 2010.

[184] HENDERSON-SELLERS, B., GONZALEZ-PEREZ, C. “A comparison of four process metamodels and the creation of a new generic standard”, *Information and Software Technology*, v. 47, n. 1, pp. 49 – 65, 2005. ISSN: 0950-5849. doi: https://doi.org/10.1016/j.infsof.2004.06.001.

[185] KUHRMANN, M., FERNÁNDEZ, D. M., STEENWEG, R. “Systematic software process development: where do we stand today?” In: *Proceedings of the 2013 International Conference on Software and System Process*, ICSSP 2013, pp. 166–170, New York, NY, USA, 2013. ACM. ISBN: 978-1-4503-2062-7. doi: 10.1145/2486046.2486077. Available at: <http://doi.acm.org/10.1145/2486046.2486077>.

[186] SADI, M. H., RAMSIN, R. “APM3: A Methodology Metamodel for Agile Project Management.” In: *SoMeT*, pp. 367–378, 2009.

[187] AYED, H., VANDEROSE, B., HABRA, N. "A metamodel-based approach for customizing and assessing agile methods". In: *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*, pp. 66–74. IEEE, 2012.

[188] CHOU, S.-C. "A process modeling language consisting of high level UML diagrams and low level process language", *Journal of Object- Oriented Programming*, v. 1, n. 4, pp. 137–163, 2002.

[189] NITTO, E. D., LAVAZZA, L., SCHIAVONI, M., et al. "Deriving executable process descriptions from UML". In: *Proceedings of the 24th International Conference on Software Engineering. ICSE 2002*, pp. 155–165, May 2002. doi: 10.1109/ICSE.2002.1007964.

[190] ISO 24744:2014. *ISO/IEC 24744:2014 - Software engineering — metamodel for development methodologies (SEMDM)*. Standard, International Organization for Standardization, Geneva, CH, Nov 2014.

[191] ENGELS, G., SAUER, S. "A Meta-Method for Defining Software Engineering Methods". In: *Graph transformations and model-driven engineering*, Springer, pp. 411–440, 2010.

[192] GONZALEZ-PEREZ, C., MCBRIDE, T., HENDERSON-SELLERS, B. "A metamodel for assessable software development methodologies", *Software Quality Journal*, v. 13, n. 2, pp. 195–214, Jun 2005. ISSN: 1573-1367. doi: 10.1007/s11219-005-6217-7. Available at: <https://doi.org/10.1007/s11219-005-6217-7>.

[193] FIRESMITH, D., HENDERSON-SELLERS, B. *The OPEN process framework: an introduction*. Addison-Wesley, 2002. ISBN: 978-0201675108.

[194] FRANCH, X., M. RIB, J. "PROMENADE: a modular approach to software process modelling and enaction", 05 1999.

[195] OMG. *Software process engineering metamodel (SPEM) 2.0 specification*. Final specification, OMG, apr 2008. http://www.omg.org/spec/SPEM/2.0/PDF/.

[196] BENDRAOU, R., GERVAIS, M.-P., BLANC, X. "UML4SPM: a UML2.0-Based metamodel for software process modelling". In: Briand, L., Williams, C. (Eds.), *Model Driven Engineering Languages and Systems*, pp. 17–38, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN: 978-3-540-32057-9.

[197] TERNITÉ, T., KUHRMANN, M. *Das V-Modell XT 1.3 Metamodell.* Standard, Technische Universität Münchenn, Germany, 2009.

[198] OMG. *Software Process Engineering Metamodel (SPEM) 2.0 Specification.* Final specification, OMG, apr 2008. http://www.omg.org/spec/SPEM/2.0/PDF/.

[199] OMG. *Object constraint language 2.4 (OCL).* Final specification, fev 2014. https://www.omg.org/spec/OCL/2.4/.

[200] FOUNDATION, T. L. *OpenAPI Specification*, 2018. Available at: <`https://www.openapis.org/`>.

[201] KOREN, I., KLAMMA, R. "The Exploitation of OpenAPI Documentation for the Generation of Web Frontends". In: *Companion Proceedings of the The Web Conference 2018*, WWW '18, p. 781–787, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN: 9781450356404. doi: 10.1145/3184558.3188740. Available at: <`https://doi.org/10.1145/3184558.3188740`>.

[202] SANDOVAL, K. *What Should You Consider Before OpenAPI Adoption?*, 2018. Available at: <`https://nordicapis.com/what-should-you-consider-before-openapi-adoption/#:~:text=OpenAPI\%20Specification\%20can\%20be\%20used,and\%20visualize\%20RESTful\%20web\%20services.`>.

[203] KOENIG, D., GLOVER, A., KING, P., et al. *Groovy in action.* Manning Publications Co., 2007.

[204] MIKOWSKI, M., POWELL, J. *Single page web applications: JavaScript end-to-end.* Manning Publications Co., 2013.

[205] VASSILIADIS, P., SIMITSIS, A., SKIADOPOULOS, S. "Conceptual modeling for ETL processes". In: *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pp. 14–21, 2002.

[206] SAJAD, M., SADIQ, M., NAVEED, K., et al. "Software Project Management: Tools assessment, Comparison and suggestions for future development", *International Journal of Computer Science and Network Security (IJCSNS)*, v. 16, n. 1, pp. 31, 2016.

[207] WARNER, J. “Thank you for 100 million repositories”. https://github.blog/2018-11-08-100m-repos/: :text=TodayNov 2018. Accessed: 2020-08-25.

[208] BLEIEL, N. “Collaborating in GitHub”. In: *2016 IEEE International Professional Communication Conference (IPCC)*, pp. 1–3. IEEE, 2016.

[209] ARORA, R., GOEL, S., MITTAL, R. K. “Supporting collaborative software development over GitHub”, *Software: Practice and Experience*, v. 47, n. 10, pp. 1393–1416, 2017.

[210] RACASAN, M. *How To Use GitHub for Agile Project Management*, May 2020. Available at: <`https://blog.zenhub.com/how-to-use-github-agile-project-management/`>.

[211] BUTLER, M., PAQUETTE, P. *Better Software & Stronger Teams - Project Management for GitHub*. ZenHub, 2016.

[212] MOCKUS, A., FIELDING, R. T., HERBSLEB, J. “A case study of open source software development: the Apache server”. In: *Proceedings of the 22nd international conference on Software engineering*, pp. 263–272, 2000.

[213] GROSSMAN, D. A., FRIEDER, O. *Information retrieval: Algorithms and heuristics*, v. 15. Springer Science & Business Media, 2012.

[214] FERNANDES, E., OLIVEIRA, J., VALE, G., et al. “A Review-Based Comparative Study of Bad Smell Detection Tools”. In: *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, EASE ’16, New York, NY, USA, 2016. Association for Computing Machinery. ISBN: 9781450336918. doi: 10.1145/2915970.2915984. Available at: <`https://doi.org/10.1145/2915970.2915984`>.

[215] PAIVA, T., DAMASCENO, A., FIGUEIREDO, E., et al. “On the evaluation of code smells and detection tools”, *Journal of Software Engineering Research and Development*, v. 5, n. 1, pp. 7, 2017.

[216] JACKSI, K. “Design and implementation of online submission and peer review system: A case study of ejournal of university of zakho”, *International Journal of Scientific and Technology Researches*, v. 4, n. 8, pp. 83–85, 2015.

[217] PEWNY, J., HOLZ, T. “EvilCoder: automated bug insertion”, *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016.

[218] DOLAN-GAVITT, B., HULIN, P., KIRDA, E., et al. "LAVA: Large-Scale Automated Vulnerability Addition", *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 110–121, 2016.

[219] BONETT, R., KAFLE, K., MORAN, K., et al. "Discovering Flaws in Security-Focused Static Analysis Tools for Android using Systematic Mutation". In: *USENIX Security Symposium*, 2018.

[220] GHALEB, A., PATTABIRAMAN, K. "How Effective Are Smart Contract Analysis Tools? Evaluating Smart Contract Static Analysis Tools Using Bug Injection". In: *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2020, p. 415–427, New York, NY, USA, 2020. Association for Computing Machinery. ISBN: 9781450380089. doi: 10.1145/3395363.3397385. Available at: <`https://doi.org/10.1145/3395363.3397385`>.

[221] EOYANG, G. H. *Conditions for self-organizing in human systems.* Union Institute, 2001.

[222] LENCIONI, P. "The five dysfunctions of a team". In: *A workshop for teams. Pfeiffer, a Wiley Imprint*, 2012.

[223] COVEY, S. R., MERRILL, R. R. *The speed of trust: The one thing that changes everything.* Simon and schuster, 2006.

[224] TOUSIGNANT, D. *Agile Maturity Matrix*, 2019. Available at: <`https://capeprojectmanagement.com/agile-self-assessment/`>.

[225] AMBLER, S. "Survey says: agile works in practice", *Dr. Dobb's Journal*, v. 31, n. 9, pp. 62–64, 2006.

[226] ANDERSON, D. J. *Kanban: successful evolutionary change for your technology business.* Blue Hole Press, 2010.

[227] AGILE, S. *Metrics - Scaled Agile Framework*, 2012. Available at: <`https://www.scaledagileframework.com/metrics/`>.

[228] AGILE, S. *Organizational Agility*, 2012. Available at: <`https://www.scaledagileframework.com/organizational-agility/`>.

[229] AGILE, S. *Team and Technical Agility*, 2012. Available at: <`https://www.scaledagileframework.com/team-and-technical-agility/`>.

[230] JOSHI, A., KALE, S., CHANDEL, S., et al. “Likert scale: Explored and explained”, *Current Journal of Applied Science and Technology*, pp. 396–403, 2015.

[231] JAMIESON, S. “Likert scales: How to (ab) use them?” *Medical education*, v. 38, n. 12, pp. 1217–1218, 2004.

[232] KUZON, W., URBANCHEK, M., MCCABE, S. “The seven deadly sins of statistical analysis”, *Annals of plastic surgery*, v. 37, pp. 265–272, 1996.

[233] STEVENS, S. S., OTHERS. “On the theory of scales of measurement”, *Science, New Series*, v. 103, pp. 677–680, Jun 1946.

[234] BORGATTA, E. F., BOHRNSTEDT, G. W. “Level of measurement: Once over again”, *Sociological Methods & Research*, v. 9, n. 2, pp. 147–160, 1980.

[235] KNAPP, T. R. “Treating ordinal scales as interval scales: an attempt to resolve the controversy”, *Nursing research*, v. 39, n. 2, pp. 121–123, 1990.

[236] CARIFIO, J., PERLA, R. “Resolving the 50-year debate around using and misusing Likert scales”, *Medical education*, v. 42, n. 12, pp. 1150–1152, 2008.

[237] HODGE, D. R., GILLESPIE, D. F. “Phrase completion scales: a better measurement approach than Likert scales?” *Journal of Social Service Research*, v. 33, n. 4, pp. 1–12, 2007.

[238] LEUNG, S.-O. “A comparison of psychometric properties and normality in 4-, 5-, 6-, and 11-point Likert scales”, *Journal of Social Service Research*, v. 37, n. 4, pp. 412–421, 2011.

[239] WU, H., LEUNG, S.-O. “Can Likert scales be treated as interval scales?—A Simulation study”, *Journal of Social Service Research*, v. 43, n. 4, pp. 527–532, 2017.

[240] MILLER, G. “Agile software development for the entire project”, *CrossTalk*, v. 18, n. 12, pp. 9–12, 2005.

[241] PIKKARAINEN, M., SALO, O., STILL, J. “Deploying Agile Practices in Organizations: A Case Study”. In: Richardson, I., Abrahamsson, P., Messnarz, R. (Eds.), *Software Process Improvement*, pp. 16–27, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN: 978-3-540-32271-9.

[242] 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.

[243] BATRA, D. "Modified agile practices for outsourced software projects", *Communications of the ACM*, v. 52, n. 9, pp. 143–148+10, 2009.

[244] POPPENDIECK, M., CUSUMANO, M. "Lean software development: a tutorial", *IEEE Software*, v. 29, n. 5, pp. 26–32, 2012.

[245] 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.

[246] GREGORY, P., BARROCA, L., TAYLOR, K., et al. *Agile challenges in practice: a thematic analysis*, v. 212. 2015.

[247] DIKERT, K., PAASIVAARA, M., LASSENIUS, C. "Challenges and success factors for large-scale agile transformations: a systematic literature review", *Journal of Systems and Software*, v. 119, pp. 87–108, 2016.

[248] 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.

[249] TERNITE, T. *Variability of Development Models.* Tese de Doutorado, Clausthal University of Technology, 2010.

# Appendix A

# Agility Assessment Approaches

**1. *42 point test: How Agile are You***
In WATERS 2008 [82], the author presented an approach composed of a 42-statement questionnaire that should be used in the following way: the Project Manager should ask every team member of an agile team (including the product owner, tester, manager, everyone) to review the statements "honestly". They should score 1 for each statement they believe they are consistent and it could be audited. Otherwise they should score 0 for the statement. The author suggested to calculate the average final score but did not provide any indication of how to analyze this result.

**2. *A Better Team***
(HERMIDA 2009 [83]) proposed an online agility assessment approach called *Abetterteam*. 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 2007 [122]. However, the author did not indicate how the questionnaire is related to the practices proposed by Shore and the rationale behind the assessment result.

**3. *A Corporate Agile 10-point Checklist***
YATZECK 2012 [72] proposed a two-checklist method to aid the adoption and assessment of 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"*. The items in the second checklist are similar to the agile smells proposed in this study since they describe practices that may jeopardize the adoption of agile methods. However, these items from the agile smells in some 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 detection of these practices may be threatened by the bias of the person performing the agility assessment who has to interpret the practice and determine how to check it; *(b)* 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; *(c)* the solution is based on checklists that have to be manually filled by the Project Manager; the items are focused on the *Scrum* method.

**4. *Agile Adoption Interview***

*Agile Adoption Interview* (BONAMASSA 2018 [84]) is a web-based survey to assist agile team members self-assess their skills in agile development and to provide information about areas of strengths and opportunities for individual improvements. The survey, which can be used to assess skill in Scrum or Kanban method, is composed of open and closed question organized in 5 sections: *Overall*, *Team Dynamics*, *Scrum/Kanban Events*, *Scrum/Kanban Intrinsics*, and *Scrum/Kanban Roles.* After submitting the responses, the tool sends to the participants an email with the assessment result.

**5. *Agile Alert***

*Agile Alert* (HOFFMANN *et al.* 2018 [85]) is a web-based assessment tool designed to aid organizations and agile teams to rate their agile capabilities. The tool is divided into 2 parts: part 1, named *"Do you work in an agile framework?"*, is organized in the sections *Agile Strategy*, *Agile structure*, and *Agile culture* while part 2, named *"Are you agile?"*, is organized in the sections *Hyperaware*, *Informed Decision-making*, and *Fast Execution.* Each section contains statements that describe a specific agile capability and that should be rated using a 5-point Likert scale that ranges from 1 to 5, where 1 means a low capability and 5 a high capability. The results should be interpreted as follows: 0–30 points: *Agile Alert!*; 30–60 points: *Agile Beginner*; 60–90 points: *Agile Adopter*; and 90–120 points:

*Agile Front-Runner.*

**6. *Agile Assessment***

In (NOWINSKI 2016 [86]), the author presented a self-assessment agility approach available as a spreadsheet survey with 66 statements/questions grouped in 7 areas: *product ownership*, *agile process*, *team*, *quality*, *engineering practices*, *fun and learning*, and *integration.* Every team member should assess each statement using a 5 point likert scale. The spreadsheet is configured to calculate the average assessment of each statement for the whole team and for each area. The average of each area is used to plot a radar diagram that graphically presents the results. One of the main limitations of this approach come from the fact that the author did not provide any reference material to aid the interpretation of the statements. Thus, the participants in the survey should assess the statement and assign a 5 point Likert scale to it only by analyzing the statement description.

**7. *Agile Enterprise Survey***

*Agile Enterprise Survey* (LEWIS and WENDLER 2016 [87]) is a web-based self-assessment survey designed by Storm Consulting in collaboration with the Dresden University. The survey's questionnaire has 44 statements and 2 open-ended questions organized in 6 sections as follows: *Values and Practices*, *Working Environment*, *Capabilities - human resources*, *Activities*, *"Blue sky" thinking*, and *Organisation background.* The survey presents a set of statements and asks the participant to specify, using a 5-point Likert scale, how well these statements reflect their organization. The section *"Blue Sky" thinking* presents two open-ended questions including one that asks "*If you could wave a magic wand to make any changes you wished to your working environment, what would you change?*". After submitting the answers, the assessment result is calculated and sent by email to the participant. The authors did not indicate how the questions where selected, how they are linked with the agile practices and how the answers are analyzed.

**8. *Agile Excellerate***

*Agile Excellerate* (SFIRLOGEA and GEORGESCU 2020 [88]) is a web-based questionnaire that assists agile team members to evaluate their understanding of agile principles, their values and adherence to good practices. It also highlights potential issues related to *trust*, *team cohesion*, *commitment*, *constructive conflicts* and *accountability.* It is based on several theories related to self-organization

(*Container-Difference-Exchange* (EOYANG 2001 [221])), team building (*Five dysfunctions of a team* (LENCIONI 2012 [222])) and trust (*Speed of trust* (COVEY and MERRILL 2006 [223])). The questionnaire is composed of 80 questions that assess the developer's opinions and perceptions about various aspects of the team agility. Questions are grouped in 8 analysis dimensions, covering the most important aspects of the Agile practice: *Respect and Communication*, *Collaborative Improvement*, *Sustainable Delivery*, *Disciplined Self-Organization*, *Predictable Quality*, *Empowered Courage*, *Focused Commitment*, and *Transparency and Visibility*. Results are consolidated and analyzed by the authors (the current version of *Agile Excellerate* does not feature automatic reporting). After analysis, the following reports are provided: 1. Radar chart of team agility based on all analysis dimensions; 2. Results on each dimension emphasizing critical aspects (low scoring or abnormal distribution of answers) 3. Correlation map (how various answers correlate or not) 4. Container — Difference — Exchange score 5. Scoring of potential team dysfunctions 6. Trust analysis: integrity, intent, capabilities and results.

**9. *Agile Health Dashboard***

*Agile Health Dashboard* (LAGESTEE 2012 [30]) is a coaching tool available as spreadsheet that helps agile teams to continuously improve their development process. To use the tool, a team member should manually fill a pre-configured sheet entering information about each sprint (start and end dates, number of completed stories, team velocity, etc). Based on the raw data provided, the tool calculates and shows a dashboard organized in 4 areas: *Sprint Planning*, *Sprint Velocity*, *Team Flow*, and *Team Dynamics*. The team should use the data emerging from the dashboard to find areas to become a stronger, more agile team.

**10. *Agile Journey Index (AJI)***

KREBS 2011 [29] proposed an agility assessment model called *Agile Journey Index (AJI)* that aids organizations in improving their application of the agile method. The model covers 19 key practices organized in 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.

**11. *Agile Maturity Assessment***

TOUSIGNANT 2019 [33] presented the *Agile Maturity Assessment*, a self-assessment approach available as a web tool that aims at measuring the organization agile maturity according to the *Agile Maturity Model* (TOUSIGNANT 2019 [224]). The approach is composed of 60 agree/disagree statements that, after completed, generate a weighted total score that indicates the organization maturity level as follows: 0 - 80 points: "*Ad-hoc Agile*"; 81-160 points: "*Doing Agile*"; 161-240 points: "*Being Agile*"; 241 - 320 points: "*Thinking Agile*"; and $>$ 320 points: "*Culturally Agile*". One of the main limitations of the tool is the lack of transparency on how the total score is calculated. The approach is a commercial tool but it is possible to run individual assessments free of charge.

**12. *Agile Skills Self-Assessment***

The BPM Institute proposed in (BPMI 2019 [89]) the *Agile Skills Self-Assessment*, a web-based survey to assist agile team members in creating a professional development game plan. The survey covers 6 critical practice areas: *1. Agile Concepts*, *2. Agile Rituals and Ceremonies*, *3. Agile Business Analysis Principles*, *4. Estimation and Velocity*, *5. Creating and Managing Quality User Stories*, and *6. Utilizing Waterfall Business Analysis Techniques in Agile*. Each area has 5 questions that should be scored using a 5-point Likert scale as follows: *1. Not at all*, *2. Somewhat*, *3. Middling*, *4. Mostly*, and *5. Very*. The final score, that ranges from 30 to 150, indicates the agile skill level (*Beginner*, *Intermediate*, *Advanced*, or *Expert*) across the 6 critical practice areas covered.

**13. *Agile Team Evaluation***

In GUNNERSON 2015 [90], the author proposed a text-based questionnaire, *Agile Team Evaluation*, to aid development teams to evaluate themselves. The questionnaire has 17 yes/no questions organized in 4 groups (*Delivery of Business Value*, *Code Health*, *Team Health*, and *Organization Health*). The author, who intended to provide a "*less prescriptive approach*", suggested questions such as "*Is the team healthy and happy?*" and "*Is the code well architected?*" that aim at promoting internal team discussions rather than defining a degree of agility to the team.

**14. *Agility Maturity Self Assessment***

There are also models that aim to assess team members individually. CAMPBELL and MACIVER 2010 [91] defined 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 on how to analyze the answers.

**15. *Agility Maturity Self Assessment Survey***

In RIBEIRO 2015 [92], the author proposed the *Agile Maturity Self-Assessment Survey*, 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 on how to analyze the answers and to assess the skill of the individuals.

**16. *Agility Questionnaire***

In BRITSCH 2017 [77], BRITSCH proposed the *Agility Questionnaire*, a spreadsheet-based questionnaire that helps to assess whether agile development is the proper approach for a specific organization and project and highlights associated challenges, risks and areas where specific tailoring is required. The *Agility Questionnaire* is not a self-assessment approach. Instead of that, the approach was designed to support consultant companies (called suppliers) and organizations willing to adopt agile development (called clients) to collaboratively assess the client's agile capability and propose the best ways to work. The questionnaire is composed of 60 questions organized in two parts: *Agility Profile* and *Project Profile*. Each question should be assigned with a 5-point agree/disagree Likert scale (*Strongly Agree*, *Agree*, *Neutral*, *Disagree*, and *Strongly Disagree*). The answers of the first part (*Agility Profile*) are organized in 6 areas: *Value Focus*, *Ceremony*, *Collaboration*, *Decisions and Information*, *Responsiveness*, and *Experience*. The results of the second part (Project Profile) are organized in 12 areas: *Confidence*, *Objectives and Goals*, *Volatility*, *Funding / Resourcing Challenge*, *Analysis Challenge*, *Political / Delivery Challenge*, *Technology Challenge*, *Design Challenge*, *Reputational Risk*, *Legal / Regulatory Risk*, *Financial Risk*, and *Operational Risk*.

**17. *Back-of-a-Napkin Agile Assessment***

The *Back-of-a-Napkin Agile Assessment* (HENDRICKSON 2008 [93]) is a text-based agile assessment checklist composed of 10 statements that aim at promoting internal team discussions rather than assessing the agility to the team.

**18. *Balbes' Agility Assessment***

BALBES 2015 [94] proposed a text-based self-assessment approach to evaluate how agile teams are improving their ability to be agile over time. The approach is composed of assessment questions that has 6 statements that correspond to different levels of maturity as described below: *Level 0: No Capability*; *Level 1: Beginning*; *Level 2: Learning*; *Level 3: Practicing*; *Level 4: Measuring*; and *Level 5: Innovating* The assessment questions are grouped into 9 different areas: *Technical Craftsmanship*, *Quality Advocacy*, *User Experience*, *Team Dynamics*, *Product Ownership*, *Project Management*, *Risk Management*, *Organizational Support*, and *Change Management.* Once an assessment is complete and responses to each question are evaluated, results can be aggregated to show progress in each area.

**19. *Borland Agile Assessment***

In (SCHUMACHER 2009 [95]), SCHUMACHER presented the *Borland Agile Assessment 2009*, a text-based survey that was initially developed to be used as a internal coaching tool. The survey consists of 12 questions answered on a 5-point agree/disagree scale. There is no "score" to this assessment that should be administered anonymously with results reported in an aggregate form. The *Borland Agile Assessment 2009* is a diagnostic tool to help development teams reflect on their processes and identify ways to improve (although the author did not make clear how to analyze the results). It should not be used to measure "improvement" from a previous assessment, only relative importance of potential improvements to their current situation. The author claimed the approach was cross-referenced with the Manifesto for Agile Development BECK *et al.* 2001 [53], as well as with agile principles from COCKBURN 2002 [23], SHORE and WARDEN 2007 [122], and AMBLER 2006 [225]. However, no evidence of such relations was provided.

**20. *Business Agility Manifesto***

*Business Agility Manifesto* (BURLTON *et al.* 2018 [96]) is a text-based assessment approach that aims to provide initial insights into an organization's need and readiness to become more agile. The approach consists of a questionnaire composed of 47 yes/no questions divided into 8 sections, *1. Perpetual Change*, *2. Business Strategy and Value Creation*, *3. Business Integrity*, *4. Business Solution Agility*, *5. Organization Agility*, *6. Value Chain Perspective*, *7. Business Knowledge and its Management*, and *8. Business Knowledge-Base / Single source of business truth.* The authors suggested the survey can be used to organizations understand their gaps to become more agile but they failed in providing details of how to analyze the survey results.

**21. *Cargo Cult Agile Checklist***

*Cargo Cult Agile Checklist* (WOLPERS 2016 [97]) is a text-based questionnaire that should be used as a start point for organizations adopting agile development to assess what part of the agile transition is going well and where action needs to be taken. The questionnaire has 25 yes/no questions that are similar to the agile smells described in this research in the sense they denote practices that may jeopardize the adoption of the agile development culture. One of the main limitations of this approach is that the author only provided the question statement. There is no further description or a hint on how to identify the occurrence of such practices. Regarding the analysis of the results, the author provided a 5-level scale ranging from "*Well done!*" (the first level with 0-2 "yes") to "You either haven't started going agile yet" (the last level with 21-25 "yes").

**22. *Comparative Agility (CA)***

In (WILLIAMS *et al.* 2010 [73]), 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, which 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 6-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.

**23. *Comprehensive Agility Measurement Tool (CAMT)***

*Comprehensive Agility Measurement Tool (CAMT)* (ERANDE and VERMA 2008 [98]) is a text-based tool that supports the assessment of an organization's level of agility. The approach proposes a unit measure, *Comprehensive Agility Index (CAI)*, that indicates the level of agility on a scale of 1 to 5, where 1 means "least agile" and 5 means "highly agile". To calculate this index, the approach uses a questionnaire that assesses 10 critical agility factors: 1. *TAKT time*; 2. *Plant Capacity*; 3. *Inventory*; 4. *Problem Solving*; 5. *e-manufacturing*; 6. *Continuous Improvement*; 7. *Operational Flexibility*; 8. *SMED / quick changeover*; 9. *Internal Customer Satisfaction*; and 10. *Human Resource Management.* Each critical factor should be assigned with a 5-point Likert scale that score from 1 to 5 points.

**24. *Depth of Kanban***

*Depth of Kanban* (ACHOUIANTZ 2013 [78]), proposed by Achouiantz, is a graph-based coaching tool (not an evaluation or compliance tool) for assessing the depth of Kanban (ANDERSON 2010 [226]) adoption in an organization. The tool is available as an offline spider graph that is structured around the 7 Kanban principles: *1. Visualize*, *2. Limit Work in Progress*, *3. Manage Flow*, *4. Make Policies Explicit*, *5. Implement Feedback Loops*, *6. Improve*, and *7. Effects.* Each axe has a different number of yes/no questions (the *Limit Work in Progress* axe, for example, has 4 questions while the *Visualize* axe has 13). The result of each axe (i.e., the level of agility) is denoted by the number of "yes" received. Regarding the analysis of the results, the author divided the spider graph into four areas (represented by different colors): *Necessary for sustainable improvements* (red), *Improving Sustainably* (yellow), *Excellent* (light green), and *Lean* (dark green). The red area on the graph defines the minimal depth a team must reach in order to start improving on its own. While the team is "in the red" it cannot improve. The other colors indicate other "levels" of depth, the greener the better.

**25. *Department of Defense Guide***

The U.S. Department of Defense (DoD) proposed in (OF DEFENSE DOD 2018 [99]) the *Defense Innovation Board Guide: Detecting Agile BS*, a text-based approach to provide guidance to DoD program executives and acquisition professionals on how to detect software projects that are really using agile development versus those that are simply waterfall or spiral development in agile clothing (*"agile-scrum-fall"*). The guide is divided into 4 sections: (a) Section 1, named *"Key flags that a project is not really agile"*, has 6 statements that may indicate a project is not using a process based on agile development; (b) Section 2 describes a set of tools usually used by agile teams; (c) Section 3 has a questionnaire organized in 5 subsections: *Questions to Ask Programming Teams* with 4 open questions, *Questions for Program Management* with 4 open questions, *Questions for Customers and Users* with 3 open questions, and *Questions for Program Leadership* with 6 open questions; and (d) Section 4 has a graphical version of the questionnaire with a flow connected through yes/no questions that illustrates the path to a desired agile development process.

**26. *Enterprise and Team Level Agility Maturity Matrix***
The *Enterprise and Team Level Agility Maturity Matrix* (ELIASSEN-GROUP 2013 [40]) 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.

**27. *Enterprise Business Agility Maturity Survey***
In (RIBEIRO 2018 [100]), RIBEIRO proposed the *Enterprise Business Agility Maturity Survey*, an approach to support organizations measure their agility capability. The survey is composed of 53 questions (including an open question) organized in 6 sections: *Leadership and Culture*, *Lean Business and Portfolio Management*, *Organisational Structure*, *Agile Mindset and Methods*, *Performance and Measurements*, and *Make It Stick and Sustain.* The author did not provide indications of how to analyze the answers and to assess the organization agility capability.

**28. *Five Key Numbers to Assess Agile Engineering Practices***
NIELSEN 2011 [101] proposed a text-based questionnaire to assess the team agile engineering practices. The questionnaire is composed of 5 questions that should be scored using a gauge scale divided into three areas: green, yellow, and red. For

example, the question "*How many manual steps does it take to get a build into production?*" has its gauge scale divided as follows: 0-1 steps: green; 2-9 steps: yellow; and 9-15 steps: red. The red area indicates the engineering practice has to be improved. The yellow area indicates the engineering practice is acceptable but it could be improved. The green area indicates the engineering practice is well implemented.

**29. *GAO's Agile Assessment Guide***

The U.S. Government Accountability Office (GAO) has published in (GAO 2020 [102]) the *Agile Assessment Guide* to aid federal agencies, departments, and auditors in assessing an organization's readiness to adopt Agile methods. The guide contains 5 text-based checklists to assess specific areas of agile adoption: *1. Adoption of Agile Methods Checklist*: 24 statements organized in 9 areas; *2. Requirements Development Checklist*: 16 statements organized in 8 areas; *3. Contracting for an Agile Program Checklist*: 9 statements organized in 3 areas; *4. Agile and Program Monitoring and Control Checklist*: 9 statements organized in 3 areas; and *5. Agile Metrics Checklist*: 14 statements organized in 6 areas.

**30. *How Agile are you? A 50 Point Test***

*How Agile are you? A 50 Point Test* (FINITE 2019 [103]) is a web-based survey to help agile teams to determine how agile they are. The survey is composed of 50 yes/no questions, allowing a team to arrive at a score out of 50 for each respondent. Every team member of the agile team, including the Product Owner, testers, and managers have to honestly answer each statement. Once each team member has completed the 50 point test, add up the points for each respondent and average them to arrive at a total score for the team. Ideally, an agile team should have an average score greater than 40 points. If a team's score is below 40, they should be looking to update your processes and team culture.

**31. *IBM DevOps Practices Self-Assessment***

*IBM DevOps Practices Self-Assessment* (IBM 2008 [104]) is an 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.

**32. *Joe's Unofficial Scrum Checklist***

LITTLE 2012 [105] adapted the approach proposed by KNIBERG 2012 [81] and proposed the *Joe's Unofficial Scrum Checklist*, an approach to assist Scrum teams to assess their agility. The approach, that should be used as basis for discussion preferably with the full team, has a checklist with 87 yes/no questions organized in 6 areas: *The Bottom Line*, *Core Scrum*, *Recommended*, *Engineering Practices*, *Scaling*, and *Positive Indicators*.

**33. *Karlskrona Test***

In (SEUFFERT 2019 [106]), SEUFFERT presented a self-assessment approach, named *Karlskrona Test*, that was developed in 2008-2009 with companies in Sweden and Germany to see how far an agile adoption came and to monitor progress over time. The test has 11 single-choice questions where each question has 4 options (2 of them score 0 and 2 score 1). The author suggested to submit the survey to all team members. The final result is calculated according to the average amount of points for the whole team and ranges from *Grade 1 - Waterlfall* to *Grade 5 - Agile*. Although the author claimed this approach is an "*easy way to claim an organization is agile*", there are no indications on how the questions are related to agile practices or empirical evidences to support this statement.

**34. *Kanban Maturity Assessment***

*Kanban Maturity Assessment* (CHIVA 2019 [107]) is a web-based assessment approach that allows managers and Kanban coaches or consultants to help the teams evaluate and understand their progress and level of understanding of principles and practices of the Kanban Method (ANDERSON 2010 [226]). The *Kanban Maturity Assessment* consists of 9 sections: Section 1 is reserved to the assessment configuration; Sections 2 to 6 focus on the 6 core Kanban practices, namely *visualize*, *limit WIP*, *manage flow*, *explicit policies*, *feedback loops*, and *improvement*; Section 8 assesses service and organizational effects of Kanban adoption; Section 9 focuses on *Fitness for Purpose*. To what extent the Service is servicing customer expectations. Each section contains a set of statement that should be answered using a 5-point Likert scale: *Strongly agree*, *Agree*, *Neutral*, *Disagree*, and *Strongly disagree*.

**35. *Lean Agile Intelligence***

The *Lean Agile Intelligence* (MCCALLA and GIFFORD 2016 [37]) is an assessment platform available as online questionnaires. The approach provides the ability of customizing out-of-the-box assessment templates or creating new questionnaires from a question bank compiled from published works of agile specialists, framework reference guides, and collaborative feedback sessions with coaches. The assessment results are presented as dashboards that aggregate team assessment results in a format that captures a holistic view of the organizations agility maturity and identifies patterns preventing the organizations from achieving their desired outcomes.

**36. *Lean/Agile Depth Assessment Checklist A3***

In (YERET 2013 [79]), YERET adapted the approach proposed by ACHOUIANTZ 2013 [78] and proposed the *Lean/Agile Depth Assessment Checklist A3*, a graph-based coaching tool for evaluating the current agile capability of a team. The tool is available as an offline spider graph that is structured around 7 perspectives: *1. Visualize  Manage the Flow*; *2. Business Value Driven Development*; *3. Individuals and Interactions Feedback Loops*; *4. Engineering Practices*; *5. Build and Deployment*; *6. Empowered Teams and Individuals*; and *7. Improve*. Each axe has a different number of yes/no questions (the *Visualize  Manage the Flow* axe, for example, has 15 questions while the *Build and Deployment* axe has 6). The result of each axe (i.e., the level of agility) is denoted by the number of "yes" received. Regarding the analysis of the results, the author divided the graph into 3 areas (represented by different colors): (a) the red area indicates that the team has to improve its capability in this perspective; (b) the yellow area indicates that the team has an acceptable capability on this perspective but there are some problems that need to be addressed; and (c) the green area indicates that the team has good capability in this perspective.

**37. *Lebow's Agile Assessment***

LEBOW 2018 [108] presented an agility self-assessment approach composed of two elements: (a) a questionnaire and (b) a checklist; The questionnaire contains 10 agility factors (*Team Communication*, *User Accessibility*, *Team Location*, *Team Structure*, *Delivery Frequency*, *Measurement of Progress*, *Ability to Change Direction*, *Testing*, *Planning Approach*, and *Process Philosophy*) that are rated from 1 to 5, where 1 being the least agile and 5 being the most agile. The final score of the questionnaire, which is the sum of the rate assigned to each agility factor, should be analyzed as follows: 50 points: *Agile maven*; 40-49 points: *Agilist all the way*; 30-39 points: *Agilist in training*; 20-29 points: *Closet agilist*; 10-19 points: *Thanks for taking the test.* The checklist, that is named "*You might not be agile if...*", has 10 statements that describe "bad" practices (i.e., practices that may impair the adoption of agile development). One of the statements, for example, is "*Your white boards are mostly white*".

**38. *Measure.Team***

*Measure.team* (ALBRECHT and EDDINGS 2020 [80]) is a web-based self-assessment survey that aids agile teams tracking their progress and monitoring improvements over time. The survey has 16 statements that should be scored using a 5-point Likert scale as follows: *Not at all/Not sure*, *Rarely*, *Sometimes*, *Often*, and *Consistently.* After submitting the questionnaire, the tool calculates and presents an overall score and, for each statement, its corresponding score and the following sections: *Why it is valuable to be consistent*, *How to start How to improve How to sustain*, and *Additional Resources.*

**39. *Nokia Test***

The *Nokia Test* (VODDE and SUTHERLAND 2010 [109]) for Scrum teams was developed originally by Bas Vodde at Nokia Siemens Networks in Finland and has been updated several times with the contribution of Jeff Sutherland. The test is a self-assessment questionnaire organized in 10 agile areas: *Iteration*, *In-Sprint Testing*, *Sprint Stories*, *Product Owner*, *Product Backlog*, *Estimation*, *Sprint Burndown*, *Retrospective*, *ScrumMaster*, and *Team.* Each area has a set of statement that should be scored by each person in a team. The total score of an area ranges from 0 to 10 and, hence, the total score ranges from 0 to 100. The team score is the average of the total score of each team member. The authors failed in providing any indication of how to analyze the team score.

**40. *Objectives Principles Strategies (OPS)***

SOUNDARARAJAN 2013 [110] proposed the *Objectives, Principles and Strategies Framework* (OPS), a framework that assists agility assessment by identifying 5 elements: 1. *Objectives* of the agile development; 2. *Principles* that support the *Objectives*; 3. *Strategies* that implement the *Principles*; 4. *Linkages* that relate *Objectives* to *Principles*, and *Principles* to *Strategies*, and 5. *Indicators* for assessing the extent to which an organization supports the implementation and effectiveness of the *Strategies*.

**41. *Open Assessments***

The *Open Assessments* (SCRUM.ORG 2020 [111]) is a series of web-based questionnaires focuses on assessing someone's knowledge on specific areas of Scrum. The series is composed of the following tests: *Scrum Open*: 30 questions to assess basic knowledge of Scrum; *Nexus Open*: 15 questions to assess basic understanding of the Nexus Framework; *Product Owner Open*: 15 questions to assess knowledge of the role of the Product Owner in Scrum; *Developer Open*: 30 questions to assess knowledge of development practices used across a Scrum Team; *Scrum with Kanban Open*: 15 questions to assess knowledge of practicing Professional Scrum with Kanban; and *Agile Leadership Open*: 10 questions to assess knowledge of Agile Leadership essentials.

**42. *Organizational Agility Self-Assessment***

The *Scaled Agile* initiative proposed 8 self-assessment approaches (AGILE 2012 [227]): 1. *Business Agility Self-Assessment*, 2. *Lean Portfolio Management Self--Assessment*, 3. *Continuous Learning Culture Self-Assessment*, 4. *Organizational Agility Self-Assessment*, 5. *Enterprise Solution Delivery Self-Assessment*, 6. *Lean Agile-Leadership Self-Assessment*, 7. *Agile Product Delivery Self-Assessment*, and 8. *Team and Technical Agility Self-Assessment*. These approaches are available as spreadsheet questionnaires where the respondents should assign a 6-point Likert scale to each question: *True* (5 points), *More True than False* (4 points), *Neither False nor True* (3 points), *More False than True* (2 points), *False* (1 point), and *Not Applicable* (0 point). The approaches more related to this research are the *Organizational Agility Self-Assessment* (AGILE 2012 [112]) and the *Team and Technical Agility Self-Assessment* (AGILE 2012 [39]). The *Organizational Agility Self-Assessment* enables organizations to assess their proficiency in the *Organizational Agility* (AGILE 2012 [228]) competency that describes how Lean-thinking people and Agile teams optimize their business processes, evolve strategy with clear and decisive new commitments, and quickly adapt the organization as needed to capitalize on new opportunities. This assessment approach is composed of 29 questions divided into 3 dimensions: *Lean Thinking People and Agile Teams*, *Lean Business Operations*, and *Strategy Agility*.

**43. *People 10 Team Assessment Approach***

In (SHOUKATH 2012 [113]), SHOUKATH presented the *People 10 Team Assessment Approach* a text-based assessment approach for organizations to benchmark the agile maturity of their teams. The approach is composed of 24 engineering practices (for example *Continuous integration*, *Refactoring*, *Build frequency*). Each practice has 2 statements, one that best describes an *'iterative'* team and another that best describes an *'agile'* team. Someone using the approach to assess a given team should mark, for each engineering practice, 1 point against the statement that best describes the team: *'iterative'* or *'agile'* team. At the end, the assessed team has 2 scores, an *'iterative'* team score and *'agile'* team score. The greater score indicates the team's strongest capability.

**44. *Perceptive Agile Measurement (PAM)***

The method proposed by SO and SCHOLL 2009 [114], the *Perceptive Agile Measurement* (PAM), is an agility assessment approach composed of 48 yes/no question organized in 8 agile areas, namely *Iteration Planning*, *Iterative Development*, *Continuous Integration and Testing*, *Stand-Up Meetings*, *Customer Access*, *Customer Acceptance Tests*, *Retrospectives*, and *Collocation*.

**45. *Quick Self-Assessment of Your Organization's Agility***

PARRY 2009 [115] defined a questionnaire to aid organizations to self-assess their agility. The questionnaire has 22 statements that should be scored using the following scale: 1 point if the statement is not true for the team; 3 points if the statement is somehow true for the team; and 5 points if the statement is completely true for the team. The author failed in describing how to analyze the final score.

**46. *Retropoly***

*Retropoly* (SFIRLOGEA and GEORGESCU 2017 [74]) is a game, based on the Monopoly game concept, to be used during retrospective meetings to aid agile teams self-assessing themselves. It is mainly designed for Scrum teams, but it is suitable with minor adjustments for any other agile methodology. The game is an alternative to traditional retrospective meetings and some benefits reported by the authors are: (a) It encourages an honest self-assessment of each member of the team and the positive feedback for the support provided by colleagues. It will improve the ability of the team to take common decisions in a timely manner and the practice of moderated debates; (b) It strengthens the team relationships by getting to know each other through sharing of personal life aspects, like hobbies and passions; and (c) It allows to observe how retrospectives are improving over the time. The game contains, among other things, a deck of 18 cards with questions about agile practices.

**47. *Scrum Assessment Series***

The *Scrum Assessment Series* (HAWKS 2013 [116]) is an agility assessment approach divided in 5 series, each focusing on a different Scrum practice: 1. *Daily Scrum*; 2. *Retrospective*; 3. *Sprint Planning*; 4. *Sprint Review*; and 5. *Release Planning*. Each series contains a questionnaire with yes/no questions organized in 3 sections (*The Basics*, *Good*, and *Awesome*) and a section *Ideas for Improvement* with statements to aid improving that area.

**48. *Scrum Checklist***

The *Scrum Checklist* (KNIBERG 2012 [81]) 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 yes/no questions 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?"*.

**49. *ScrumMaster Checklist***

The *ScrumMaster Checklist* (JAMES 2007 [117]) is a text-based coaching tool for Scrum Masters elaborated according to the personal experience of the author that has a large experience training Scrum Masters. The approach is divided into 2 parts. The first part contains 42 one-choice questions while the second part contains open questions to describe the organizational impediment. The questions in the first part are organized in 4 parts: 1. *How Is My Product Owner Doing?*; 2. *How Is My Team Doing?*; 3. *How Are Our Engineering Practices Doing?*, and 4. *How Is The Organization Doing?*. Each question should be marked with one of the following options: *Option 1* (if the respondent considering they are "doing well"); *Option 2* (for "could be improved and I know how to start"); *Option 3* (for "could be improved, but how?"); or *N/A* (for "not applicable" or "would provide no benefit"). The author did not give indications on how to calculate and analyze the results. The instructions provided in the approach indicate that if the respondents check off most of the items, they are on track to become an efficient Scrum Master.

**50. *Self Assessment Tool for Transitioning to Agile***
ROTHMAN 2013 [118] proposed the *Self Assessment Tool for Transitioning to Agile*, a self-assessment tool for measuring agile maturity composed of 8 questions. The author also supplied, for some question, the expected answers for those organizations willing to adopt agile development and a discussion about the answer. As an example, the question: "*If you are doing iterations, are they four weeks or less? The answer should be yes. Many of us like one or two week iterations. Why? Because you get feedback more often rather than less often. And, you get to see working software*".

**51. *Squad Health Check Model***
The *Squad Health Check Model* (KNIBERG and LINDWALL 2014 [119]) is a game-based approach used to aid organizations tracking the health of their squads (the term used by the authors to denote a small, cross-functional, and self-organizing development team). The model, that was firstly developed and applied at Spotify, prescribes three phases: *Phase 1.* A Workshop to collect data where members of a squad discuss and assess their current situation based on a number of different perspectives, namely *Delivery Value*, *Easy to release*, *Fun*, *Health of Codebase*, *Learning*, *Mission*, *Pawns or Players*, *Speed*, *Suitable Process*, *Support*, and *Teamwork*. This phase is supported by a deck of cards where each card has 2 statements, one green describing a good aspect of the assessed perspective and one red describing a bad aspect. For each perspective, the team has to define a colour that best describes their current squad for that perspective where: (a) *Green* means the squad is satisfied with their ability on that perspective and does not see need for improvement now; (b) *Yellow* means there are some important problems that need to be addressed; and (c) *Red* means the perspective needs to be improved. *Phase 2.* Create a graphical summary of the result. *Phase 3.* Use the data to help the squads improve.

**52. *Team and Technical Agility Self-Assessment***
*Team and Technical Agility* (AGILE 2012 [229]) competency which is the collection of foundation practices on which Agile development is based (see approach 42).

This assessment approach has 34 questions divided into 3 dimensions: *Agile Teams*, *Team of Agile teams*, and *Built-in Quality*. Regarding the analysis of the results, the approaches calculate the average score for each dimension considering the questions with a positive score (*Not applicable* answers are not counted in the final result) and present a radar chart with the dimensions and their corresponding score. However, there is no clear indication of how to interpret these results and which action should be taken.

**53. *Team Barometer***

*Team Barometer* (JANLÉN 2014 [76]) is an approach that has a twofold goal: (a) to evaluate how an agile team gets stronger over time, and (b) to be conducted as an alternative to the traditional iteration retrospective meetings. The approach is executed as a survey in a workshop with the whole agile team. The survey consists of 16 team characteristics, packaged as a deck of cards. Each card has a headline naming the corresponding characteristic of the team, a green and a red statement. The green statement denotes a good practice while the red statement denotes a bad practice. For example, the card that corresponds to the characteristic "Trust" has the following statements: "*We have the courage to be honest with each other. We don't hesitate to engage in constructive conflicts*" (green) and "*Members rarely speak their mind. We avoid conflicts. Discussions are tentative and polite.*" (red). Team members vote green, yellow or red for each card in the meeting. Green means that the member agrees with the green statement, red that the member agrees with the red statement. A yellow vote means that the member thinks it is neither green nor red but something in the middle. Once all cards have been run through, the team reflects and discusses the results.

**54. *TeamMetrics***

The *TeamMetrics* (VERWIJS 2017 [120]) is a web-based survey proposed by Christiaan Verwijs that aims at helping agile teams improve by gathering data about key team factors such as *team morale*, *motivation*, *happiness*, *learning*, *performance*, *communication*, and *leadership* and interpret the results with the help of benchmarks. The survey has 10 statements that should be scored using a 19-point Likert scale where the lowest value means *Very inaccurate* and the highest value means *Very accurate*. One of the statements that compose the survey, for example, is "*My job requires me to use a number of high level or complex skills*".

**55. *Test Maturity Card Game***

The *Test Maturity Card Game* (SCHOOTS and SCHUURKES 2017 [121]) is game-based tool designed to help teams assess and improve their testing capability. The approach is supported by a card game that helps teams discuss and identify strengths and weaknesses in their process. The model consists of a set of criteria organized in 6 different areas: *1. Test Culture*, *2. Context*, *3. Trait*, *4. Skills*, *5. Processes*, and *6. Artefacts*. The team uses a card game to identify the criteria most relevant to their context. The approach is supported by a card game that is used to aid the teams identify the most relevant criteria to their context and to assess the teams ability in the selected criteria.

**56. *The Agile Self-Assessment Game***

LINDERS 2019 [75] presented the *Agile Self-Assessment Game*, a game-based approach that assists teams to reflect on their own team interworking, discover how agile they are and decide what they can do to increase their agility. The game consists of a deck of cards with statements on applying agile practices organized in 5 "suits": 52 *Basic Agile* cards, 39 *Scrum* cards, 52 *Kanban* cards, 26 *DevOps* cards, and 26 *Business Agility* cards.

**57. *The Art of Agile Development***

In (SHORE and WARDEN 2007 [122]), SHORE and WARDEN proposed a self-assessment survey that aims to help agile teams review and evaluate their approach to adopting agile development. It focuses on five important aspects of agile development: *Thinking*, *Collaborating*, *Releasing*, *Planning*, and *Developing*. The approach is available as a text-based survey composed of 46 yes/no questions. Each question has a specific weight that ranges from 3 to 75. The final score, that is calculated as the sum of each question, should be analyzed as follows: 75 points or less: "*immediate improvement required*" (red); 75 to 96 points: "*improvement necessary*" (yellow); 97, 98, or 99: "*improvement possible*" (green); and 100: "*no further improvement needed*".

**58. *The Joel Test: 12 Steps to Better Code***

SPOLSKY proposed in (SPOLSKY 2000 [123]) a self-assessment questionnaire for measuring how good a software team is. The questionnaire has 12 yes/no questions. Each 'yes' answer scores 1 point and the final score, which is the sum of the 12 questions, should be analyzed as follows: A score of 12 means the organization is perfect, 11 is tolerable, but 10 or lower and the organization has serious problems.

**59. *Visual Management Self-Assessment***

The *Visual Management Self-Assessment* (HOGAN 2017 [124]) is a web-based self-assessment survey to aid organization to identify what techniques an organization are currently doing and find next steps for improvement. The survey is useful as a baseline for measuring an organization depth of Kanban adoption over time and also as a checklist of ideas for techniques to try. The survey is organized in 3 areas: *Clarity* of the the work (position of the work, performance measures and identification of problems), *Controls* (over team capacity and commitments to stakeholders) and *Collaboration* (feedback mechanisms and team collaboration practices). The tool sends to all participants a report on insights into the state of Kanban across the survey once there are enough responses.

**60. *Yodiz's Team Agility Self Assessment***

The *Yodiz's Team Agility Self Assessment* (YODIZ 2017 [125]) is a self-assessment survey available as a spreadsheet that can be used to support agile teams understanding whether and in which extend they are applying agile practices. The survey is composed of 37 questions organized in 8 agile areas (*Team*, *Backlog*, *Daily Scrum*, *Sprint*, *Coding Practices*, *Testing*, *Business*, and *Retrospective*). Each question assesses whether the team is applying a specific agile practice and is scored using the following scale: *0 points: Never*, *1 point: Rarely*, *2 points: Occasionally*, *3 points: Often*, *4 points: Very Often*, and *5 points: Always.* The findings from the survey are illustrated in the form of a pie chart. The chart visualizes the overall progress and where the team needs to improve.

# Appendix B

# Survey Design and Data Analysis

In Chapter 4 we presented the methodology conducted to produce the *Catalogue of Agile Smells*. In a nutshell, the catalogue was elaborated through three steps: 1. Step 1: We conducted a literature review to identify the agile smells; 2. Step 2: We conducted a survey to reveal the relevance of the agile smells; and 3. Step 3: We organized the agile smells as a catalogue. This appendix focus on discussing some aspects of the design of the survey and the analysis of the collected data.

The remainder of this Appendix is divided in three issues:

1. Asymmetric Likert scale;

2. Likert scale as interval data; and

3. Aggregation of two Likert scales.

**Asymmetric Likert scale**

The first issue is related to the design of the Likert scale used in the survey. As we presented in Section 4.4, the questionnaire was composed of two questions that accepted the following answers:

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

This scale is unbalanced and asymmetric since it has 1 negative option (*Not relevant*) and 3 positive options (*Slightly relevant*, *Very relevant*, and *Absolutely relevant*). As the scale has more positive options than negative, it may induce the respondent to give a positive answer. As suggested by JOSHI *et al.* 2015 [230], it is preferable to use a symmetric Likert scale since it provides independence to a participant to choose any response in a balanced and symmetric way in either directions.

### Likert scale as interval data

The second issue discussed in this appendix regarding the decision of using the data collected from a Likert scale, which is an ordinal scale, as continuous data. The treatment as continuous data was done when we add the answers of the SQ1 and SQ2 and when we used the result to rank the agile smells (see Figures 4.3 and 4.4). Theoretically, only interval and ratio scales are considered to be continuous, where arithmetic operations can be conducted, while nominal and ordinal scales are considered to be categorical data, where arithmetic operation should not be applied. There are many studies dealing with the disadvantages of treating ordinal as interval scales. JAMIESON 2004 [231] reviewed ways of using Likert scales, and stated that it is a common practice, but controversial, to treat a Likert scale as interval scale. Computing means and standard deviations for Likert scale data are considered to be inappropriate. Instead, nonparametric statistics should be used. KUZON *et al.* 1996 [232] maintained that one of the seven deadly sins of statistical analysis is using parametric analysis for ordinal scales. There are thus arguments against the use of Likert scales as continuous measures. On the other hand, there are arguments in favor of considering Likert scales as continuous interval scales. STEVENS *et al.* 1946 [233] accepted, under some circumstances, the use of ordinal as interval scales and this was re-stated in other studies such as BORGATTA and BOHRNSTEDT, KNAPP 1980, 1990 [234, 235], CARIFIO and PERLA 2008 [236]. There have been numerous studies using Likert scale as interval data. The impasse is then that, even though the Likert scale violates basic statistical assumptions, many studies find it useful. One alternative to mitigate this threat is to increase the number of Likert scale points to make it closer to continuous scales and normality HODGE and GILLESPIE 2007 [237], LEUNG 2011 [238]. Traditionally, the number of points in a Likert scale can be as few as four or five (in this study we used four) but if it can be increased to eleven, a common metric that ranges from 0 to 10, as recommended by HODGE and GILLESPIE 2007 [237], LEUNG 2011 [238], it can be treated as a continuous measure and hence arithmetic operations can be used WU and LEUNG 2017 [239].

### Aggregation of two Likert scales

As presented in Section 4.4.1, the questionnaire used in the survey is composed of two questions (***Survey-RQ1*** and ***Survey-RQ2***) and each question has an associated value that varies from 0 to 3 ((a) *Not relevant (0 pts)* (b) *Slightly relevant*

*(1 pt)* (c) *Very relevant (2 pts)* (d) *Absolutely relevant (3 pts))*. The relevance of a given agile smell (that is ultimately used to rank the agile smells) is calculated by adding the values of the two answers as shown in Figure B.1.

$$\boxed{relevanceByParticipant(p) = answerQuestion1(p) + answerQuestion2(p)}$$

Figure B.1: Formula of agile smell relevance by participant.

We understand that analysing these two questions together to generate a unique classification may jeopardize the resulting ranking since each question assesses a different aspect of an agile smell and therefore the sum of the collected data should not be applied. Thus, in the remainder of this section, we present an alternative analysis that considers the questions separately.

Firstly, we analyzed the question ***Survey-RQ1*** and calculated the value *relevanceQ1* as shown in Figure B.2. Table B.1 shows the agile smells ranked according to the value of *relevanceQ1* that each agile smells achieved.

Secondly, we analyzed the question ***Survey-RQ2*** and calculated the value *relevanceQ2* as shown in Figure B.3. Table B.2 shows the agile smells ranked according to the value of *relevanceQ2* that each agile smells achieved.

$$\boxed{relevanceQ1 = \sum_{p=p1}^{pn} answerQuestion1(p)}$$

Figure B.2: Formula of final agile smell relevance for question 1.

$$\boxed{relevanceQ2 = \sum_{p=p1}^{pn} answerQuestion2(p)}$$

Figure B.3: Formula of final agile smell relevance for question 2.

Table B.1: Agile smells ranked by their relevance to agility assessment (question 1 of the survey).

| **R** | **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 | 1 | 2 | 3 | 3 | 2 | 2 | 3 | 2 | 3 | 1 | 1 | 2 | 3 | 3 | 2 | 3 | 2 | 2 | 3 | **46** |
| 02 | *Absence of Frequent Deliveries* | 3 | 1 | 2 | 3 | 2 | 2 | 3 | 3 | 3 | 2 | 1 | 1 | 2 | 3 | 2 | 2 | 3 | 2 | 1 | 3 | **44** |
| 03 | *Goals Not Defined or Poorly Defined* | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 2 | 2 | 1 | 2 | 2 | 2 | 3 | 3 | 2 | 3 | 2 | **44** |
| 04 | *Iteration Without a Deliverable* | 2 | 2 | 2 | 2 | 3 | 2 | 3 | 3 | 1 | 2 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 2 | 3 | 1 | **43** |
| 05 | *Complex Tasks* | 2 | 1 | 3 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 2 | 3 | 2 | 2 | 2 | **42** |
| 06 | *Iteration Without an Iteration Planning* | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 1 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 2 | **41** |
| 07 | *Iteration Without an Iteration Retrospective* | 2 | 2 | 2 | 3 | 1 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 3 | 1 | 2 | 3 | 2 | 2 | 2 | **40** |
| 08 | *Iteration Started without an Estimated Effort* | 2 | 2 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 2 | 3 | 1 | 2 | 3 | 3 | 2 | 3 | 2 | 3 | 2 | **39** |
| 09 | *Iteration Without an Iteration Review* | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 3 | 2 | 2 | 3 | 2 | 3 | 2 | **39** |
| 10 | *Shared Developers* | 3 | 2 | 3 | 2 | 2 | 2 | 1 | 1 | 0 | 1 | 3 | 1 | 1 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | **39** |
| 11 | *Absence of Timeboxed Iteration* | 2 | 1 | 2 | 2 | 3 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 3 | 2 | 2 | 3 | **38** |
| 12 | *Unplanned Work* | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 3 | 2 | **37** |
| 13 | *Dependence on Internal Specialists* | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 1 | 1 | 3 | 2 | 3 | 1 | **36** |
| 14 | *Unfinished Work in a Closed Iteration* | 2 | 1 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 3 | 3 | 3 | 2 | 2 | 2 | 3 | **36** |
| 15 | *Absence of Timeboxed Meeting* | 2 | 2 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 2 | 2 | 1 | 2 | **33** |
| 16 | *Absence of Test-driven Development* | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 0 | 2 | 2 | 1 | 1 | 2 | 3 | 0 | 3 | 3 | 2 | 2 | 3 | **33** |
| 17 | *Large Development Team* | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 2 | 1 | 2 | 3 | 2 | 1 | 3 | 2 | 1 | 2 | **30** |
| 18 | *Long Break Between Iterations* | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 0 | 1 | 1 | 2 | 1 | 1 | 3 | 2 | 1 | 3 | 2 | 3 | 2 | **30** |
| 19 | *Concurrent Iterations* | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 3 | 2 | 3 | 1 | **27** |
| 20 | *Iterations with Different Duration* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 2 | 1 | 3 | 1 | 1 | 3 | 0 | 0 | 2 | 2 | 1 | 2 | **25** |

Table B.2: Agile smells ranked by their identification strategy relevance (question 2 of the survey).

| **R** | **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 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 2 | 3 | 2 | 1 | 3 | 3 | 2 | 2 | 3 | 2 | 2 | 3 | **50** |
| 02 | *Absence of Frequent Deliveries* | 3 | 1 | 3 | 3 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 1 | 3 | 3 | 3 | 2 | 3 | 2 | 1 | 3 | **46** |
| 03 | *Iteration Without a Deliverable* | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 3 | 1 | 1 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 1 | **43** |
| 04 | *Iteration Without an Iteration Planning* | 2 | 1 | 2 | 1 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 1 | 1 | 2 | 2 | 2 | 3 | 2 | 3 | 2 | **40** |
| 05 | *Goals Not Defined or Poorly Defined* | 3 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 2 | 3 | 2 | 3 | 2 | **38** |
| 06 | *Absence of Timeboxed Iteration* | 3 | 2 | 2 | 2 | 2 | 3 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 3 | 2 | 2 | 3 | **38** |
| 07 | *Iteration Without an Iteration Retrospective* | 2 | 1 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 1 | 1 | 3 | 2 | 2 | 1 | **37** |
| 08 | *Complex Tasks* | 3 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | **36** |
| 09 | *Iteration Started without an Estimated Effort* | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 3 | 3 | 2 | 3 | 2 | 3 | 2 | **36** |
| 10 | *Iteration Without an Iteration Review* | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 3 | 1 | 2 | 3 | 2 | 3 | 2 | **35** |
| 11 | *Unplanned Work* | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 2 | 3 | 2 | 3 | 2 | **33** |
| 12 | *Dependence on Internal Specialists* | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 3 | 1 | 2 | 3 | 2 | 3 | 2 | **33** |
| 13 | *Unfinished Work in a Closed Iteration* | 3 | 1 | 2 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 2 | 3 | 3 | 3 | 2 | 2 | 2 | 3 | **32** |
| 14 | *Shared Developers* | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 3 | 1 | 1 | 3 | 1 | 2 | 2 | 2 | 3 | 3 | **31** |
| 15 | *Absence of Timeboxed Meeting* | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | **31** |
| 16 | *Absence of Test-driven Development* | 2 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 2 | 1 | 1 | 1 | 3 | 0 | 3 | 3 | 2 | 2 | 3 | **29** |
| 17 | *Large Development Team* | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 2 | 3 | 2 | 2 | 3 | 2 | 1 | 3 | **27** |
| 18 | *Long Break Between Iterations* | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 2 | 1 | 1 | 3 | 2 | 2 | 3 | 2 | 3 | 2 | **27** |
| 19 | *Iterations with Different Duration* | 2 | 1 | 1 | 2 | 1 | 0 | 1 | 0 | 1 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | **26** |
| 20 | *Concurrent Iterations* | 1 | 0 | 0 | 0 | 1 | 2 | 1 | 1 | 1 | 0 | 2 | 2 | 0 | 0 | 1 | 1 | 3 | 2 | 3 | 1 | **22** |

### Discussion

In this appendix, we discussed and presented possible solutions for three issues related to the survey conducted in this research (see Chapter 4). First, we analysed the problem of using an asymmetric Likert scale that could be fixed by using a symmetric Likert scale. Then, we discussed the problem of treating the data collected from a Likert scale, which is an ordinal scale, as continuous data. One possible solution to mitigate this threat is increasing the number of points of the Likert scale. The third threat is related to the analysis of the collected data that aggregated two Likert scales. An alternative analysis that considers the two questions separately was presented. Although these items are important, they were were pointed by reviewers at an advanced stage of this research so we were not able to apply the proposed solutions during the catalogue elaboration. However, we strongly suggest that researchers consider these points in an eventual continuation or replication of this research.

# Appendix C

# Agile smells literature review selected studies

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

**P1:** SCHWABER, K. SCRUM development process. In: *Business Object Design and Implementation*, Springer London, pp. 117–134, 1997 [58].

**P2:** STAPLETON, J. *Dynamic systems development method: the method in practice.* Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 1997. ISBN: 0201178893 [63].

**P3:** BECK, K. Embracing change with extreme programming, *Computer*, v. 32, n. 10, pp. 70–77, 1999 [55].

**P4:** BECK, K. *Extreme programming explained: embrace change.* Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 2000. ISBN: 0-201-61641-6 [55].

**P5:** CUNNINGHAM, W. Extreme programming. http://www.-extremeprogramming.org/, 1999. Accessed: 2017-12-01 [57].

**P6:** LUCA, J. D. Feature driven development FDD. http://www.-featuredrivendevelopment.com/, 1999. Accessed: 2017-12-01 [62].

**P7:** HIGHSMITH, III, J. A. *Adaptive software development: a collaborative approach to managing complex systems.* New York, NY, USA, Dorset House Publishing Co., Inc., 2000. ISBN: 0-932633-40-4 [24].

**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, Washington, DC, USA, 2001. IEEE Computer Society [144].

**P9:** PALMER, S. R., FELSING, M. *A practical guide to feature-driven development.* Pearson Education, 2001. ISBN: 0130676152 [].

**P10:** MAURER, F., MARTEL, S. Extreme programming: rapid development for web-based applications, *IEEE Internet Computing*, v. 6, n. 1, pp. 86–90, 2002 [149].

**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 [150].

**P12:** LINDVALL, M., BASILI, V. R., BOEHM, B. W., et al. 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*, London, UK, UK, 2002. Springer-Verlag [145].

**P13:** ABRAHAMSSON, P., SALO, O., RONKAINEN, J., et al. *Agile software development methods - review and analysis.* Relatório Técnico 478, VTT Publications, Espoo, Finland, 2002 [64].

**P14:** SCHWABER, K., BEEDLE, M. *Agile software development with scrum.* 1st ed. Upper Saddle River, NJ, USA, Prentice Hall PTR, 2001. ISBN: 0130676349 [59].

**P15:** COCKBURN, A. *Agile software development.* Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN: 0-201-69969-9 [23].

**P16:** MARTIN, R. C. *Agile software development: principles, patterns, and practices.* Upper Saddle River, NJ, USA, Prentice Hall PTR, 2003. ISBN: 0135974445 [151].

**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. doi: 10.1109/INMIC.2004.1492915 [166].

**P18:** MCMAHON, P. Extending agile methods: a distributed project and organizational improvement perspective, *CrossTalk*, , n. 5, pp. 16–19, 2005 [172].

**P19:** MILLER, G. Agile software development for the entire project, *CrossTalk*, v. 18, n. 12, pp. 9–12, 2005 [240].

**P20:** PIKKARAINEN, M., SALO, O., STILL, J. Deploying Agile Practices in Organizations: A Case Study. In: Richardson, I., Abrahamsson, P., Messnarz, R. (Eds.), *Software Process Improvement*, pp. 16–27, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN: 978-3-540-32271-9 [241].

**P21:** AMBLER, S. Survey says: agile works in practice, *Dr. Dobb's Journal*, v. 31, n. 9, pp. 62–64, 2006 [225].

**P22:** CAO, L., RAMESH, B. Agile software development: ad hoc practices or sound principles? *IT professional*, v. 9, n. 2, pp. 41–47, 2007 [168].

**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 [169].

**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 [242].

**P25:** BATRA, D. Modified agile practices for outsourced software projects, *Communications of the ACM*, v. 52, n. 9, pp. 143–148+10, 2009 [243].

**P26:** MISRA, S. C., KUMAR, V., KUMAR, U. Identifying some important success factors in adopting agile software development practices, *J. Syst. Softw.*, v. 82, n. 11, pp. 1869–1890, nov 2009 [170].

**P27:** LI, J. Research and practice of agile unified process. In: *ICSTE 2010 - 2010 2nd International Conference on Software Technology and Engineering, Proceedings*, v. 2, pp. V2340–V2343, 2010 [171].

**P28:** WILLIAMS, L. Agile software development methodologies and practices. In: *Advances in Computers*, v. 80, Elsevier, pp. 1–44, 2010 [152].

**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 [66].

**P30:** SHI, Z., CHEN, L., CHEN, T.-E. Agile planning and development methods. In: *ICCRD2011 - 2011 3rd International Conference on Computer Research and Development*, v. 1, pp. 488–491, 2011 [173].

**P31:** POPPENDIECK, M., CUSUMANO, M. Lean software development: a tutorial, *IEEE Software*, v. 29, n. 5, pp. 26–32, 2012 [244].

**P32:** SLETHOLT, M., HANNAY, J., PFAHL, D., et al. What do we know about scientific software development's agile practices? *Computing in Science and Engineering*, v. 14, n. 2, pp. 24–36, 2012 [159].

**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 [245].

**P34:** ALZOABI, Z. Agile software: body of knowledge. In: *Business Intelligence and Agile Methodologies for Knowledge-Based Organizations: Cross-Disciplinary Applications*, IGI Global, pp. 14–34, 2012 [153].

**P35:** YATZECK, E. A corporate agile 10-point checklist. http://pagilista.blogspot.com/2012/12/a-corporate-agile-10-point-checklist.html, Dec 2012. Accessed: 2019-06-30 [72].

**P36:** MEYER, B. *Agile!: the good, the hype and the ugly*, v. 9783319051550, *Agile!: the good, the hype and the ugly.* Cham, Spring, 2014 [4].

**P37:** PAPATHEOCHAROUS, E., ANDREOU, A. Empirical evidence and state of practice of software agile teams, *Journal of Software: Evolution and Process*, v. 26, n. 9, pp. 855–866, 2014 [160].

**P38:** CHAGAS, L. F., DE CARVALHO, D. D., LIMA, ADAILTON MAGALHÃES AN REIS, C. A. L. Systematic Literature Review on the Characteristics of Agile Project Management in the Context of Maturity Models. In: Mitasiunas, A., Rout, T., O'Connor, R. V., et al. (Eds.), *Software Process Improvement and Capability Determination*, pp. 177–189, Cham, 2014. Springer International Publishing. ISBN: 978-3-319-13036-1 [167].

**P39:** MORAN, A. Agile software development. In: *Agile Risk Management*, Springer International Publishing, pp. 1–16, Cham, 2014 [154].

**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 [155].

**P41:** BERTEIG, M. Rules of scrum. http://www.agileadvice.com/rules-of-scrum/, 2015. Accessed: 2017-12-01 [60].

**P42:** GHANI, I., BELLO, M. Agile adoption in IT organizations, *KSII Transactions on Internet and Information Systems*, v. 9, n. 8, pp. 3231–3248, 2015 [156].

**P43:** GREGORY, P., BARROCA, L., TAYLOR, K., et al. *Agile challenges in practice: a thematic analysis*, v. 212. 2015 [246].

**P44:** ELORANTA, V.-P., KOSKIMIES, K., MIKKONEN, T. Exploring ScrumBut – An Empirical Study of Scrum Anti-Patterns, *Information and Software Technology*, v. 74, 12 2015. doi: 10.1016/j.infsof.2015.12.003 [17].

**P45:** DIKERT, K., PAASIVAARA, M., LASSENIUS, C. Challenges and success factors for large-scale agile transformations: a systematic literature review, *Journal of Systems and Software*, v. 119, pp. 87–108, 2016 [247].

**P46:** UIKEY, N., SUMAN, U. Tailoring for agile methodologies: a framework for sustaining quality and productivity, *International Journal of Business Information Systems*, v. 23, n. 4, pp. 432–455, 2016 [161].

**P47:** TRIPP, J., ARMSTRONG, D. Agile methodologies: organizational adoption motives, tailoring, and performance, *Journal of Computer Information Systems*, v. 58, pp. 1–10, 10 2016. doi: 10.1080/08874417.2016.1220240 [5].

**P48:** KROPP, M., MEIER, A., BIDDLE, R. Agile Practices, Collaboration and Experience. In: Abrahamsson, P., Jedlitschka, A., Nguyen Duc, A., et al. (Eds.), *Product-Focused Software Process Improvement*, pp. 416–431, Cham, 2016. Springer International Publishing. ISBN: 978-3-319-49094-6 [162].

**P49:** JAIN, R., SUMAN, U. Effectiveness of agile practices in global software development, *International Journal of Grid and Distributed Computing*, v. 9, n. 10, pp. 231–248, 2016 [163].

**P50:** ALLIANCE, S. Learn about scrum. https://www.scrumalliance.-org/why-scrum, 2016. Accessed: 2017-12-01 [21].

**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 [248].

**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 [157].

**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*, v. 7, n. 3, pp. 167–180, 2017 [158].

**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*, v. 2018-June, pp. 1–6, 2018 [164].

**P55:** VALLON, R., DA SILVA ESTÁCIO, B., PRIKLADNICKI, R., et al. Systematic literature review on agile practices in global software development, *Information and Software Technology*, v. 96, pp. 161–180, 2018 [165].

# Appendix D

# Catalogue of Agile Smells

Table D.1: AS 01: *Lower Priority Tasks Executed First*

| **Name**: **AS 01 -** ***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**: [58], [63], [57], [55], [56], [62], [24], [61], [149], [64], [150], [59], [23], [151], [153], [4], [154], [156], [17], [21], [157], [158], [164]. |

Table D.2: AS 02: *Absence of Frequent Deliveries*

| **Name**: **AS 02 - *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**: [58], [63], [57], [55], [56], [62], [24], [61], [149], [64], [150], [59], [23], [151], [152], [66], [153], [154], [155], [156], [60], [21], [157], [158]. |

Table D.3: AS 03: *Iteration Without a Deliverable*

| **Name**: **AS 03 - *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**: [58], [63], [57], [55], [56], [62], [24], [61], [149], [64], [150], [59], [23], [151], [66], [153], [154], [156], [60], [21], [158]. |

Table D.4: AS 04: *Goals Not Defined or Poorly Defined*

| **Name**: **AS 04 - *Goals Not Defined or Poorly 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 or Poorly 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**: [58], [63], [62], [24], [61], [144], [64], [59], [23], [151], [153], [156], [17], [21]. [157]. |

Table D.5: 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 iterations. 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 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 (called Pre-Iteration Planning) a team representative (usually the most experienced) and a business analyst discuss details about the candidates iteration backlog items. 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 a 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**: [58], [63], [57], [55], [56], [62], [24], [61], [149], [64], [150], [59], [23], [151], [172], [169], [170], [152], [66], [173], [159], [153], [4], [167], [154], [155], [60], [21], [161], [162], [163], [5], [157], [164], [165]. |

Table D.6: AS 06: *Complex Tasks*

| **Name**: **AS 06 - *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**: [58], [59], [151], [166], [152], [66], [4], [154], [21], [157], [165]. |

Table D.7: AS 07: *Iteration Without an Iteration Retrospective*

| **Name**: **AS 07 - *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**: [58], [63], [62], [24], [61], [64], [59], [23], [151], [152], [159], [153] [4], [154], [155], [156], [60], [161], [5], [162], [163], [157], [158], [164], [165]. |

Table D.8: AS 08: *Absence of Timeboxed Iteration*

| **Name**: **AS 08 - *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**: [58], [24], [144], [59], [23], [151], [159], [153], [4], [154], [155], [60], [21], [158]. |

Table D.9: AS 09: *Iteration Started without an Estimated Effort*

| **Name**: **AS 09 - *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**: [58], [63], [62], [24], [61], [64], [59], [23], [151], [152], [173], [159], [153], [4], [154], [60], [21], [161], [163], [5], [157], [158], [164]. |

Table D.10: AS 10: *Iteration Without an Iteration Review*

| **Name**: **AS 10 - *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**: [58], [63], [62], [24], [61], [64], [59], [23], [151], [152], [173], [159], [153], [4], [154], [60], [21], [161], [163], [5], [157], [158], [164]. |

# Appendix E

# Software metamodel literature review selected studies

The 14 studies selected for full consideration in the literature review are listed below.

**P1:** FRANCH, X., M. RIB, J. PROMENADE: a modular approach to software process modelling and enaction, 05 1999 [194].

**P2:** CHOU, S.-C. A process modeling language consisting of high level UML diagrams and low level process language, *Journal of Object- Oriented Programming*, v. 1, n. 4, pp. 137–163, 2002 [188].

**P3:** NITTO, E. D., LAVAZZA, L., SCHIAVONI, M., et al. Deriving executable process descriptions from UML. In: *Proceedings of the 24th International Conference on Software Engineering. ICSE 2002*, pp. 155–165, May 2002. doi: 10.1109/ICSE.2002.1007964 [189].

**P4:** FIRESMITH, D., HENDERSON-SELLERS, B. *The OPEN process framework: an introduction.* Addison-Wesley, 2002. ISBN: 978-0201675108 [193].

**P5:** HENDERSON-SELLERS, B., GONZALEZ-PEREZ, C. A comparison of four process metamodels and the creation of a new generic standard, *Information and Software Technology*, v. 47, n. 1, pp. 49 – 65, 2005. ISSN: 0950-5849. doi: https://doi.org/10.1016/j.infsof.2004.06.001 [184].

**P6:** GONZALEZ-PEREZ, C., MCBRIDE, T., HENDERSON-SELLERS, B. A metamodel for assessable software development methodologies, *Software Quality Journal*, v. 13, n. 2, pp. 195–214, Jun 2005. ISSN: 1573-1367. doi: 10.1007/s11219-005-6217-7. Available at: <`https://doi.org/10.1007/s11219-005-6217-7`> [24].

**P7:** BENDRAOU, R., GERVAIS, M.-P., BLANC, X. UML4SPM: a UML2.0-Based metamodel for software process modelling. In: Briand, L., Williams, C. (Eds.), *Model Driven Engineering Languages and Systems*, pp. 17–38, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN: 978-3-540-32057-9 [196].

**P8:** OMG. *Software process engineering metamodel (SPEM) 2.0 specification.* Final specification, OMG, apr 2008. http://www.omg.org/spec/SPEM/2.0/PDF/ [195].

**P9:** SADI, M. H., RAMSIN, R. APM3: A Methodology Metamodel for Agile Project Management. In: *SoMeT*, pp. 367–378, 2009 [186].

**P10:** TERNITÉ, T., KUHRMANN, M. *Das V-Modell XT 1.3 Metamodell.* Standard, Technische Universität Münchenn, Germany, 2009 [197].

**P11:** ENGELS, G., SAUER, S. A Meta-Method for Defining Software Engineering Methods. In: *Graph transformations and model-driven engineering*, Springer, pp. 411–440, 2010 [191].

**P12:** TERNITE, T. *Variability of Development Models.* Tese de Doutorado, Clausthal University of Technology, 2010 [249].

**P13:** AYED, H., VANDEROSE, B., HABRA, N. A metamodel-based approach for customizing and assessing agile methods. In: *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*, pp. 66–74. IEEE, 2012 [187].

**P14:** ISO 24744:2014. *ISO/IEC 24744:2014 - Software engineering — metamodel for development methodologies (SEMDM).* Standard, International Organization for Standardization, Geneva, CH, Nov 2014 [190].

# Appendix F

# *Journal Submission System* Project Structure - Tasks and Iterations

The *Journal Submission System* project has a total of 111 tasks distributed over 7 iterations as shown in Table F.1.

Table F.1: Milestones and Issues of the project *Journal Submission System*

| | Id | Issue title | Labels | Status | Complexity |
|---|---|---|---|---|---|
| 1 | **Milestone 1:** Sprint 01 - Setting Up the Project - **Number of issues:** 9<br>**Status:** Closed - **Start:** Jan 1, 2020 - **Due by:** Jan 10, 2020 - **Duration:** 10 days<br>**Goal:** This sprint has as the main goal defining the software architecture and the tools that will support the remainder of the project. | | | | |
| 2 | #50 | Plan sprint 01 | [sprint planning] | done | 8 |
| 3 | #51 | Define software architecture | [architecture] | done | 40 |
| 4 | #52 | Configure CI/CD pipelines | [devops] | done | 21 |
| 5 | #53 | Configure developers environment | [devops] | done | 21 |
| 6 | #54 | Present software architecture to developers | [architecture] | done | 5 |
| 7 | #55 | Present software architecture to stakeholders | [architecture] | done | 2 |
| 8 | #56 | Create database schema | [devops] | done | 2 |
| 9 | #57 | Configure application server profiles | [devops] | done | 3 |
| 10 | #60 | Retrospective sprint 01 | [sprint retrospective] | done | 2 |
| 11 | **Milestone 2:** Sprint 02 - Authors Registration - **Number of issues:** 10<br>**Status:** Closed - **Start:** Jan 11, 2020 - **Due by:** Jan 25, 2020 - **Duration:** 15 days<br>**Goal:** - | | | | |

**Continued on next page**

| Continuation of Table F.1 | | | | | |
|---|---|---|---|---|---|
| | **Id** | **Issue title** | **Labels** | **Status** | **Complexity** |
| 12 | #71 | Write specification for the feature *Author Registration* | [req], [priority:high] | done | 8 |
| 13 | #72 | Code the feature *Author Registration* | [dev], [priorirty:high] | done | 13 |
| 14 | #73 | Code the feature *Send Confirmation Account Email to Author* | [dev], [priority:high] | done | 5 |
| 15 | #74 | Test the feature *Author Registration* | [qa], [priorirty:high] | done | 5 |
| 16 | #75 | Write specification for the feature *Integration with the ORCID platform* | [req], [priority:normal] | done | 3 |
| 17 | #76 | Code the feature *Integration with the ORCID platform* | [dev], [priority:normal] | done | 8 |
| 18 | #77 | Test the feature *Integration with the ORCID platform* | [qa], [priority:normal] | done | 5 |
| 19 | #78 | Deploy Release v.02 | [deploy] | done | 1 |
| 20 | #79 | Review sprint 02 | [sprint review] | done | 1 |
| 21 | #80 | Retrospective sprint 02 | [sprint retrospective] | done | 2 |
| 22 | **Milestone 3:** Sprint 03 - Manuscript Submission - **Number of issues:** 15<br>**Status:** Closed - **Start:** Jan 26, 2020 - **Due by:** Feb 14, 2020 - **Duration:** 20 days<br>**Goal:** This sprint aims at developing all features to support the submission of a manuscript. It includes the following features: manuscript submission form; upload documents; send an email notification to Editor; send an email notification to Authors; | | | | |
| 23 | #90 | Plan sprint 03 | [sprint planning] | done | 8 |
| 24 | #91 | Write specification for the feature *Manuscript Submission* | [req], [priority:high] | done | 5 |
| 25 | #92 | Code the feature *Manuscript Submission* | [dev], [priority:high] | done | 21 |
| 26 | #93 | Test the feature *Manuscript Submission* | [qa], [priority:high] | done | 8 |
| 27 | #94 | Write specification for the feature *Manage Categories* | [req], [priority:normal] | done | 5 |
| 28 | #95 | Code the feature *Manage Categories* | [dev], [priority:normal] | done | 8 |
| 29 | #96 | Test the feature *Manage Categories* | [qa], [priority:normal] | done | 5 |
| 30 | #97 | Code the feature *Upload documents* | [dev], [priority:normal] | done | 5 |
| 31 | #98 | Code the feature *Send Notification Email to Editor* | [dev], [priority:normal] | done | 5 |
| 32 | #99 | Code change in the feature *Author Registration*: Add new field (Second email address) | [change], [dev], [priority:low] | done | 5 |

**Continued on next page**

| Continuation of Table F.1 | | | | | |
|---|---|---|---|---|---|
| | **Id** | **Issue title** | **Labels** | **Status** | **Complexity** |
| 33 | #100 | Test change in the feature *Author Registration*: Add new field (Second email address) | [change], [qa], [priority:low] | done | 3 |
| 34 | #101 | Deploy release v.03 | [deploy] | done | 1 |
| 35 | #102 | Review sprint 03 | [sprint review] | done | 1 |
| 36 | #104 | Fix bug 001 in the feature *Send Confirmation Account Email to Author* | [bug], [dev], [priority:high] | done | 5 |
| 37 | #105 | Test Fix bug 001 in the feature *Send Confirmation Account Email to Author* | [bug], [qa], [priority:high] | done | 3 |
| 38 | **Milestone 4:** Sprint 04 - Invite Reviewers - **Number of issues:** 21<br>**Status:** Closed - **Start:** Feb 15, 2020 - **Due by:** Feb 29, 2020 - **Duration:** 15 days<br>**Goal:** This sprint aims at delivering all the features that support an Editor invite Reviewers to assess a manuscript. | | | | |
| 39 | #110 | Plan sprint 04 | [sprint planning] | done | 8 |
| 40 | #111 | Write specification for the feature *Invite Reviewers* | [req], [priority:high] | done | 8 |
| 41 | #112 | Code the feature *Find Candidate Reviewers by Manuscript Key Words* | [dev], [priority:high] | done | 8 |
| 42 | #113 | Code the feature *Send Invitation to Review Email to Reviewer* | [dev], [priority:high] | done | 5 |
| 43 | #114 | Test the feature *Invite Reviewers* | [qa], [priority:high] | done | 5 |
| 44 | #115 | Write specification for the feature *Manage Reviewers* | [req], [priority:normal] | done | 5 |
| 45 | #116 | Code the feature *Manage Reviewers* | [dev], [priority:normal] | done | 8 |
| 46 | #117 | Test the feature *Manage Reviewers* | [qa], [priority:normal] | done | 5 |
| 47 | #118 | Write specification for the feature *Integration with the Publons platform* | [req], [priority:normal] | done | 8 |
| 48 | #119 | Code the feature *Integration with the Publons platform* | [dev], [priority:normal] | done | 13 |
| 49 | #120 | Test the feature *Integration with the Publons platform* | [qa], [priority:normal] | done | 5 |
| 50 | #121 | Write specification for the feature *Manage Invitations* | [req], [priority:normal] | done | 8 |
| 51 | #122 | Code the feature *Manage Invitations* | [dev], [priority:normal] | done | 8 |
| 52 | #123 | Test the feature *Manage Invitations* | [qa], [priority:normal] | done | 5 |

**Continued on next page**

**Continuation of Table F.1**

| | Id | Issue title | Labels | Status | Complexity |
|---|---|---|---|---|---|
| 53 | #124 | Code change in the feature *Manuscript Submission*: Add new field (Agreement confirmation checkbox) | [change], [dev], [priority:low] | done | 5 |
| 54 | #125 | Test change in the feature *Manuscript Submission*: Add new field (Agreement confirmation checkbox) | [change], [qa], [priority:low] | done | 3 |
| 55 | #128 | Retrospective sprint 04 | [sprint retrospective] | done | 2 |
| 56 | #129 | Fix bug 002 in the feature *Manuscript Submission* | [bug], [dev], [priority:high] | done | 3 |
| 57 | #130 | Test Fix bug 002 in the feature *Manuscript Submission* | [bug], [qa], [priority:high] | done | 2 |
| 58 | #131 | Fix bug 003 in the feature *Send Notification Email to Editor* | [bug], [dev], [priority:normal] | done | 3 |
| 59 | #132 | Test Fix bug 003 in the feature *Send Notification Email to Editor* | [bug], [qa], [priority:normal] | done | 2 |
| 60 | **Milestone 5:** Sprint 05 - Reply Invitation to Review - **Number of issues:** 21<br>**Status:** Open - **Start:** Mar 1, 2020 - **Due by:** Mar 13, 2020 - **Duration:** 13 days<br>**Goal:** Sprint to implement Reply Invitation to Review. | | | | |
| 61 | #140 | Plan sprint 05 | [sprint planning] | done | - |
| 62 | #141 | Code refactoring in the feature *Manage Authors*: fixing critical code smells | [dev], [priority:high] | done | - |
| 63 | #142 | Test refactoring in the feature *Manage Authors*: fixing critical code smells | [qa], [priority:high] | doing | 5 |
| 64 | #143 | Write specification for the feature *Reply Invitation to Review* | [req], [priority:high] | done | - |
| 65 | #144 | Code the feature *Send Friendly Reminder Notification Email to Reviewers* | [dev], [priority:high] | to do | 5 |
| 66 | #145 | Code the feature *Reply Invitation to Review (Agree)* | [dev], [priority:normal] | to do | 8 |
| 67 | #146 | Code the feature *Reply Invitation to Review (Decline)* | [dev], [priority:normal] | to do | 5 |
| 68 | #147 | Code the feature *Send Notification Email to Editor* | [dev], [priority:normal] | to do | 5 |
| 69 | #148 | Code the feature *Send Confirmation Email to Reviewer* | [dev], [priority:normal] | to do | 13 |
| 70 | #149 | Test the feature *Reply Invitation to Review* | [qa], [priority:normal] | to do | 5 |

**Continued on next page**

| Continuation of Table F.1 | | | | | |
|---|---|---|---|---|---|
| | **Id** | **Issue title** | **Labels** | **Status** | **Complexity** |
| 71 | #150 | Code change in the feature *Send Notification Email to Editor*: Include authors' email name | [change], [dev], [priority:low] | to do | 3 |
| 72 | #151 | Test change in the feature *Send Notification Email to Editor*: Include authors' email name | [change], [qa], [priority:low] | to do | 2 |
| 73 | #152 | Deploy release v.05 | [deploy] | to do | 1 |
| 74 | #153 | Review sprint 05 | [sprint review] | to do | 2 |
| 75 | #154 | Retrospective sprint 05 | [sprint retrospective] | to do | 2 |
| 76 | #155 | Fix bug 004 in the feature *Manage Invitations* | [bug], [dev], [priority:high] | to do | 5 |
| 77 | #156 | Test Fix bug 004 in the feature *Manage Invitations* | [bug], [qa], [priority:high] | to do | 3 |
| 78 | #157 | Fix bug 005 in the feature *Manage Reviewers* | [bug], [dev], [priority:normal] | doing | 3 |
| 79 | #158 | Test Fix bug 005 in the feature *Manage Reviewers* | [bug], [qa], [priority:normal] | to do | 3 |
| 80 | #159 | Fix bug 006 in the feature *Integration with the Publons platform* | [bug], [dev], [priority:low] | to do | 3 |
| 81 | #160 | Test Fix bug 006 in the feature *Integration with the Publons platform* | [bug], [qa], [priority:low] | to do | 2 |
| 82 | **Milestone 6:** Sprint 06 - Manuscript Review - **Number of issues:** 21<br>**Status:** Open - **Start:** Mar 11, 2020 - **Due by:** Mar 17, 2020 - **Duration:** 17 days<br>**Goal:** The main goal of this sprint is to support the Reviewers to assess a manuscript. | | | | |
| 83 | #171 | Write specification for the feature *Enter Manuscript Reviews* | [req], [priority:high] | to do | 5 |
| 84 | #172 | Code the feature *Enter Manuscript Reviews* | [dev], [priority:high] | to do | 13 |
| 85 | #173 | Test the feature *Enter Manuscript Reviews* | [qa], [priority:high] | to do | 5 |
| 86 | #174 | Code the feature *Send Notification Email to Editor* | [dev], [priority:normal] | to do | 5 |
| 87 | #175 | Write specification for the feature *Send Final Result Email to Authors* | [req], [priority:normal] | to do | 5 |
| 88 | #176 | Code the feature *Send Final Result Email to Authors* | [dev], [priority:normal] | to do | 8 |

**Continued on next page**

| Continuation of Table F.1 | | | | | |
|---|---|---|---|---|---|
| | **Id** | **Issue title** | **Labels** | **Status** | **Complexity** |
| 89 | #177 | Test the feature *Send Final Result Email to Authors* | [qa], [priority:normal] | to do | 5 |
| 90 | #178 | Code change in the feature *Send Friendly Reminder Notification Email to Reviewers*: include the Editor's email | [change], [dev], [priority:low] | to do | 8 |
| 91 | #179 | Test change in the feature *Send Friendly Reminder Notification Email to Reviewers*: include the Editor's email | [change], [qa], [priority:low] | to do | 5 |
| 92 | #180 | Code change in the feature *Manage Invitations*: include filter by status | [change], [dev], [priority:low] | to do | 5 |
| 93 | #181 | Test change in the feature *Manage Invitations*: include filter by status | [change], [qa], [priority:low] | to do | 3 |
| 94 | #182 | Code refactoring in the feature *Reply Invitation to Review*: fixing critical code smells | [dev], [priority:normal] | doing | - |
| 95 | #183 | Test refactoring in the feature *Reply Invitation to Review*: fixing critical code smells | [qa], [priority:normal] | to do | 3 |
| 96 | #184 | Deploy release v.06 | [deploy] | to do | 1 |
| 97 | #185 | Review sprint 06 | [sprint review] | to do | 1 |
| 98 | #187 | Fix bug 007 in feature *Send Invitation to Review Email to Reviewer* | [bug], [dev], [priority:high] | to do | 5 |
| 99 | #188 | Test Fix bug 007 in the feature *Send Invitation to Review Email to Reviewer* | [bug], [qa], [priority:high] | to do | 3 |
| 100 | #189 | Fix bug 008 in the feature *Reply Invitation to Review (Decline)* | [bug], [dev], [priority:high] | to do | - |
| 101 | #190 | Test Fix bug 008 in the feature *Reply Invitation to Review (Decline)* | [bug], [qa], [priority:high] | to do | - |
| 102 | #191 | Fix bug 009 in the feature *Send Confirmation Email to Reviewer* | [bug], [dev], [priority:normal] | to do | - |
| 103 | #192 | Test Fix bug 009 in the feature *Send Confirmation Email to Reviewer* | [bug], [qa], [priority:normal] | to do | - |
| 104 | **Milestone 7:** Sprint 07 - Authors Notification - **Number of issues:** 12<br>**Status:** Open - **Start:** Mar 28, 2020 - **Due by:** Apr 11, 2020 - **Duration:** 15 days<br>**Goal:** The main goal of this sprint is to support the Reviewers to assess a manuscript. | | | | |
| 105 | #200 | Plan sprint 07 | [sprint planning] | done | 8 |
| **Continued on next page** | | | | | |

**Continuation of Table F.1**

| | Id | Issue title | Labels | Status | Complexity |
|---|---|---|---|---|---|
| 106 | #201 | Write specification for the feature *Authors Notification* | [req], [priority:high] | done | 8 |
| 107 | #202 | Code the feature *Authors Notification* | [dev], [priority:high] | doing | 13 |
| 108 | #203 | Test the feature *Authors Notification* | [qa], [priority:high] | to do | 5 |
| 109 | #204 | Code change in the feature *Send Final Result Email to Authors*: include review notes | [change], [dev], [priority:normal] | to do | - |
| 110 | #205 | Test change in the feature *Send Final Result Email to Authors*: include review notes | [change], [qa], [priority:normal] | to do | - |
| 111 | #206 | Code change in the feature *Enter Manuscript Reviews*: include section *Confidential Comments to the Editor* | [change], [dev], [priority:low] | to do | 8 |
| 112 | #207 | Test change in the feature *Enter Manuscript Reviews*: include section *Confidential Comments to the Editor* | [change], [qa], [priority:low] | to do | 5 |
| 113 | #208 | Code refactoring in the feature *Send Notification Email to Editor*: fixing critical code smells | [dev], [priority:normal] | to do | 8 |
| 114 | #209 | Test refactoring in the feature *Send Notification Email to Editor*: fixing critical code smells | [qa], [priority:normal] | to do | 5 |
| 115 | #211 | Review sprint 07 | [sprint review] | to do | 1 |
| 116 | #212 | Retrospective sprint 07 | [sprint retrospective] | to do | 2 |

**End of Table F.1**

# Appendix G

# *Terminal Operational System* Project Structure - Tasks and Iterations

The subset of the project structure selected for the validation conducted in this research has a total of 90 tasks distributed over 2 iterations as shown in Table G.1.

Table G.1: Sprints and their respective tasks of the project *Terminal Operational System* selected to the case study (E.E.: Estimated Effort, R.E.: Real effort, A.T.: Assigned to)

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| **Sprint #102 - Number of tasks:** 44<br>**Status:** Closed - **Duration:** 26 calendar days / 18 business days<br>**Goal:** EDI Log-In | | | | | | | | |
| 15150 | Reuniao | Done | 8,00 | 8,00 | - | - | Média | Planejamento Sprint 102 |
| 15157 | Desenv | Done | 4,00 | 1,50 | - | Não Planejada | Alta | [2.10.1] [ EDI ] Implementar EDI para armador Log-In |
| 15459 | Desenv | Done | 1,00 | 1,00 | - | Planejada | Imediata | [2.10.0] [BOLETIM CARGA DESCARGA] Alteração na query de geração de XML de Contêiner |
| 15460 | Desenv | Done | 1,00 | 0,70 | - | Planejada | Alta | [2.10.0] [PRESENÇA CARGA] Item 3 - Nova coluna na tela de filtros para inclusão |
| 15461 | Desenv | Done | 6,00 | 10,00 | - | Planejada | Normal | [2.10.1] [PRESENÇA CARGA] Item 26 - Edição Presença Carga |
| 15462 | Desenv | Done | 0,50 | 0,50 | - | Planejada | Alta | [2.10.0] [PRESENÇA CARGA] Alterar nome do menu |
| 15463 | Desenv | Done | 0,50 | 0,25 | - | Planejada | Alta | [2.10.0] [PRESENÇA CARGA] Alterar título da tela |
| 15465 | Desenv | Done | 2,00 | 1,00 | - | Planejada | Alta | [2.10.0] [PRESENÇA CARGA] Extrato do Despacho |
| 15466 | Desenv | To Test | 1,00 | 1,50 | - | Erro | Alta | [2.10.0] [PRESENÇA CARGA] Mensagem anexos errada |
| 15467 | Desenv | Done | 0,50 | 0,20 | - | Planejada | Alta | [2.10.0] [PRESENÇA CARGA] Alterar título da tela de detalhes |
| 15468 | Desenv | Done | 1,00 | 0,50 | - | Planejada | Alta | [2.10.0] [PRESENÇA CARGA] Item 15 - Indicador Parte |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 15471 | Desenv | Done | 5,00 | 3,00 | - | Planejada | Alta | [2.10.0] [PRESENÇA CARGA] Consulta inclusao de carga está retornando bookings a mais |
| 15480 | Test | Done | 1,00 | 0 | Analist 1 | Planejada | Normal | [2.10.0] [BOLETIM CARGA DESCARGA] Alteração na query de geração de XML de Contêiner |
| 15481 | Test | Done | 1,00 | 1,00 | Analist 1 | Planejada | Normal | [2.10.0] [PRESENÇA CARGA] Item 3 - Nova coluna na tela de filtros para inclusão |
| 15482 | Test | Done | 0,50 | 0,50 | Analist 1 | Planejada | Normal | [2.10.0] [PRESENÇA CARGA] Alterar nome do menu |
| 15483 | Test | Done | 0,50 | 0,50 | Analist 1 | Planejada | Normal | [2.10.0] [PRESENÇA CARGA] Alterar título da tela |
| 15486 | Test | Done | 0,50 | 0,50 | Analist 1 | Planejada | Normal | [2.10.0] [PRESENÇA CARGA] Alterar título da tela de detalhes |
| 15487 | Test | Done | 1,00 | 0,50 | Analist 1 | Planejada | Normal | [2.10.0] [PRESENÇA CARGA] Item 15 - Indicador Parte |
| 15488 | Test | Done | 5,00 | 12,00 | Analist 1 | Planejada | Normal | [2.10.0] [PRESENÇA CARGA] Consulta inclusao de carga está retornando bookings a mais |
| 15490 | Test | Done | 6,00 | 3,00 | Analist 1 | Planejada | Normal | [2.10.1] [PRESENÇA CARGA] Item 26 - Edição Presença Carga |
| 15496 | Desenv | Done | 1,00 | 3,00 | - | Planejada | Normal | [2.10.1] [PRESENÇA CARGA] Query Tela detalhes Inclusão |
| 15497 | Test | Done | - | 1,00 | Analist 1 | Planejada | Normal | [2.10.1] [PRESENÇA CARGA] Query Tela detalhes Inclusão |
| 15498 | Desenv | Done | 1,00 | 1,00 | - | Planejada | Alta | [2.10.1] [RECEPÇÃO] Liberação de Pagamento |
| 15499 | Test | Done | - | 3,00 | Analist 1 | Planejada | Normal | [2.10.1] [RECEPÇÃO] Liberação de Pagamento |
| 15502 | Desenv | To Test | 4,00 | 10,50 | - | Planejada | Normal | [2.10.1] [PRESENÇA CARGA] Nova Dinâmica/Regras dos Anexos |
| 15509 | Desenv | Done | 1,00 | 1,00 | - | Erro | Normal | [Crosscheck Descarga] [2.10.1] - O sistema deve considerar somente os CE's de importação para o crosscheck |
| 15514 | Desenv | Done | 2,00 | 1,00 | - | Planejada | Normal | [2.10.1] [PRESENÇA CARGA] Item 26 - Edição |
| 15515 | Test | Done | - | 1,00 | Analist 1 | Planejada | Normal | [2.10.1] [PRESENÇA CARGA] Item 26 - Edição |
| 15522 | Desenv | Done | 3,00 | 4,90 | - | Planejada | Muito Alta | [2.10.1] [RECEPÇÃO] [BOLETO] Nome Arquivo Remessa |
| 15524 | AD | Done | - | 1,00 | - | Planejada | Normal | [Presença de Carga] Alterações na tabela tb_presenca_carga_patio |

**Continued on next page**

| Continuation of Table G.1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Id** | **Type** | **Status** | **E.E.** | **R.E.** | **A.T.** | **Category** | **Priority** | **Title** |
| 15525 | Desenv | Done | - | 0,25 | - | Planejada | Muito Alta | [2.10.1] [PRESENÇA CARGA] Alteração no Job de atualização das presenças de carga |
| 15528 | Desenv | Done | 5,00 | 9,00 | - | Planejada | Muito Alta | [2.10.1] [PRESENÇA CARGA] Erro Siscomex |
| 15529 | Test | Done | - | 3,50 | Analist 1 | Planejada | Muito Alta | [2.10.1] [PRESENÇA CARGA] Erro Siscomex |
| 15530 | Test | Done | - | 1,00 | Analist 1 | Planejada | Muito Alta | [2.10.1] [PRESENÇA CARGA] Alteração no Job de atualização das presenças de carga |
| 15534 | Desenv | Done | 4,00 | 3,00 | - | Não Planejada | Muito Alta | [2.10.1] [RECEPCAO] Falha ao reprogramar alguns cntrs Pendentes de Aprovação |
| 15535 | Desenv | Done | 3,00 | 4,50 | - | Não Planejada | Alta | [ EDI ] [2.10.1] Validar navio através do Código Lloyd |
| 15540 | Desenv | To Test | 1,00 | 0,50 | - | Não Planejada | Normal | [2.10.1] [Editar Presença Carga] - Ajuste no status ao excluir associação de uma presença carga |
| 15557 | Desenv | To Test | 1,00 | 1,00 | - | Erro | Normal | [2.10.1] [PRESENÇA CARGA] Nova Dinâmica/Regras dos Anexos - AJUSTAR FLUXO QUANDO LOCAL = REDEX |
| 15559 | Desenv | To Test | 0,50 | 0,75 | - | Erro | Normal | [2.10.1] [PRESENÇA CARGA] Edição na tela de detalhes da CONSULTA |
| 15562 | Desenv | To Test | 2,00 | 1,00 | - | Erro | Normal | [2.10.1] [Presença de Carga] - Ajustes na tela de consulta - Detalhes |
| 15565 | Desenv | Done | 1,00 | 1,00 | - | Não Planejada | Alta | [ EDI ] [2.10.1] Importador Login - Desconsiderar segmento FTX após segmento DGS |
| 15566 | Desenv | Done | 3,00 | 1,00 | - | Erro | Normal | [PRESENCA CARGA INCLUIR] - Sistema esta salvando presencas de carga com transitodiretoats incorreto. |
| 15578 | Desenv | To Test | 3,00 | 1,00 | - | Erro | Normal | [Excluir Associação Presenca Carga] - Sistema não realiza desassociação de presença de carga. |
| 15876 | Reuniao | Done | 8,00 | 8,00 | - | - | Média | Apresentação da aplicação para cliente |
| **Sprint #103 - Number of tasks:** 46 | | | | | | | | |
| **Status:** Closed - **Duration:** 33 calendar days / 20 business days | | | | | | | | |
| **Goal:** Corretiva Liberação Pagto pedido IMO | | | | | | | | |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 15150 | Reuniao | Done | 8,00 | 8,00 | - | - | Média | Planejamento Sprint 102 |
| 15285 | Reuniao | Done | 8,00 | 8,00 | - | - | Média | [2.10.2] [ Cadastro de Booking ] Planejamento Sprint 103 |
| 13495 | Desenv | Done | 3,00 | 10,50 | - | Não Planejada | Média | [2.10.2] [ Cadastro de Booking ] Adicionar campo Terminal de Descarga |
| 14687 | Test | Done | 11,00 | 12,00 | Tester 1 | Planejada | Normal | [TRUNK] [CONTROLE DE CARGAS EXPORTAÇÃO] CANCELAR PRESENCA |
| 15166 | Test | Done | - | 0,50 | Tester 1 | Erro | Alta | [2.10.2] [LIBERAÇÃO EMBARQUE] - Sistema não está preenchendo a flag inliberacaoembarque ao liberar embarque. |
| 15419 | Test | Done | - | 9,00 | Tester 1 | Erro | Normal | [TRUNK] [Cancelar Presenca Carga] - Sistema não envia 'ADD HOLD' no cancelamento da presença de carga. |
| 15454 | Desenv | Done | 2,00 | 2,00 | Dev 2 | Erro | Normal | [TRUNK] [IMPORTAÇÃO SISCOMEX] Divergência quanto ao comportamento dos bloqueios da escala siscomex nas telas de Crosscheck de Descarga e Controle de cargas. |
| 15485 | Test | Done | 1,00 | 1,50 | - | Erro | Alta | [2.10.2] [PRESENÇA CARGA] Mensagem anexos errada |
| 15489 | Test | Done | - | 4,00 | Tester 2 | Não Planejada | Normal | [2.10.2] [ EDI ] Implementar EDI para armador Log-In |
| 15518 | Desenv | Done | 3,00 | 1,00 | - | Planejada | Alta | [2.10.2] [PRESENÇA CARGA] Item 14 - Inserir novas colunas |
| 15519 | Test | Done | - | 0,50 | Analist 1 | Planejada | Alta | [2.10.2] [PRESENÇA CARGA] Item 14 - Inserir novas colunas |
| 15520 | Test | Done | - | 1,00 | Analist 1 | - | Alta | [ 2.10.2] [ Cadastro de Booking ] Adicionar campo Terminal de Descarga |
| 15523 | Test | Done | - | 0,50 | Analist 1 | Planejada | Alta | [2.10.2] [RECEPÇÃO] [BOLETO] Nome Arquivo Remessa |
| 15536 | Desenv | Done | 8,00 | 7,50 | - | Não Planejada | Alta | [2.10.2] [Presença de carga] - Mudança de requisito na inclusão / edição da presença de carga |
| 15537 | Test | Done | - | 17,50 | - | Planejada | Alta | [2.10.2] [PRESENÇA CARGA] Nova Dinâmica/Regras dos Anexos |
| 15560 | Desenv | Done | - | 0 | Dev 3 | Erro | Alta | [2.10.2] [LIBERACAO SISCOMEX DESCARGA ] Sistema não envia Lacres para o Navis |
| 15561 | Test | Done | - | 4,50 | Tester 1 | Erro | Alta | [2.10.2] [LIBERACAO SISCOMEX DESCARGA ] Sistema não envia Lacres para o Navis |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 15567 | Test | Done | - | 1,00 | - | Erro | Alta | [2.10.2] [PRESENCA CARGA INCLUIR] - Sistema esta salvando presencas de carga com transitodiretoats incorreto. |
| 15570 | Desenv | Done | 4,00 | 24,50 | Dev 1 | Não Planejada | Alta | [2.10.2] [Presença de Carga] - Máscara para os tipos de documento |
| 15571 | Desenv | To Test | 3,00 | 2,00 | - | Erro | Alta | [2.10.2] [Presença de Carga] Ajustes na tela de detalhes |
| 15574 | Desenv | Done | 2,00 | 1,00 | - | Erro | Normal | [TRUNK] [CONTROLE CARGA] Presenças de carga com status -7 - Associação Excluída- estão consideradas -cargas- para liberação de embarque |
| 15579 | Test | Done | - | 0,50 | - | Erro | Alta | [2.10.2] [Excluir Associação Presenca Carga] - Sistema não realiza desassociação de presença de carga. |
| 15580 | Desenv | Done | 1,00 | 1,00 | - | Erro | Normal | [2.10.2] [Liberar Embarque] Sistema exibe mensagem de Erro ao liberar presenca de carga |
| 15581 | Test | Done | - | 1,00 | Tester 1 | Erro | Normal | [2.10.2] [Liberar Embarque] Sistema exibe mensagem de Erro ao liberar presenca de carga |
| 15582 | Desenv | To Test | 4,00 | 4,00 | Dev 3 | Não Planejada | Alta | [2.10.2] [ EDI ] Enviar código lloyd para o Portal de Monitoramento |
| 15583 | Desenv | Done | 1,00 | 0,25 | - | Erro | Normal | [TRUNK] [Controle de cargas] Icone das ocorrências divergentes entre o grid de documentos e o icone das cargas |
| 15585 | Desenv | To Test | 1,00 | 1,50 | - | Erro | Normal | [2.10.2] [Presença de carga] Exibição incorreta de anexo na tela de detalhes - consulta |
| 15587 | Desenv | Done | 2,00 | 2,00 | Dev 1 | Não Planejada | Normal | Auxílio ao testador em Controle Documental |
| 15588 | Desenv | Done | 1,50 | 1,50 | Dev 1 | Não Planejada | Normal | Configuração máquina: Baixar e atualizar projetos |
| 15590 | Test | Done | - | 8,00 | - | Planejada | Alta | [2.10.2] [RECEPÇÃO - STATUS] Verificar status para carga perigosa |
| 15591 | Test | Done | - | 1,50 | - | Não Planejada | Normal | [2.10.2] [ EDI ] Enviar código lloyd para o Portal de Monitoramento |
| 15593 | Desenv | Done | 1,00 | 1,00 | - | Erro | Alta | [2.10.2] [Boleto Cobranca Log Arquivo de Remessa] Adicionar Appender e log |
| 15594 | Test | Done | - | 0,50 | Analist 1 | Erro | Alta | [2.10.2] [Boleto Cobranca Log Arquivo de Remessa] Adicionar Appender e log |
| 15595 | Desenv | Done | 3,00 | 3,00 | - | Erro | Normal | [CONTROLE CARGAS] - Sistema esta enviando 2 HOLDS no cancelamento da presença de carga. |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 15597 | Test | Done | - | 5,00 | Tester 1 | Erro | Normal | [CONTROLE CARGAS] - Sistema esta enviando 2 HOLDS no cancelamento da presença de carga. |
| 15598 | Desenv | Done | 3,00 | 3,00 | - | Planejada | Muito Alta | [2.10.2] [RECEPÇÃO IMO - Liberar Pagamento] Ajustes |
| 15599 | Test | Done | - | 1,00 | Analist 1 | Planejada | Alta | [2.10.2] [RECEPÇÃO IMO] Ajustes |
| 15607 | Test | Done | - | 1,00 | Analist 1 | Planejada | Alta | [2.10.2] [LIBERAR PAGAMENTO] Pedido Recepção Reefer |
| 15634 | Desenv | To Test | 2,00 | 2,00 | - | Não Planejada | Muito Alta | [2.10.2] [Recepção] Edição de Cliente/Pagador |
| 15643 | Desenv | Done | 1,00 | 1,00 | Dev 1 | Planejada | Muito Alta | [2.10.2] [PRESENÇA CARGA] Item 35 - Mensagem Sucesso |
| 15644 | Test | Done | 1,00 | 1,00 | Analist 1 | Planejada | Alta | [2.10.2] [PRESENÇA CARGA] Item 35 - Mensagem Sucesso |
| 15651 | Test | Done | - | 4,00 | Tester 1 | Não Planejada | Normal | Geração de massa para teste/homologação |
| 15654 | Test | Done | - | 1,50 | Tester 1 | Não Planejada | Alta | [2.10.2] [Presença de carga] [Granel] Mudança de requisito na inclusão / edição da presença de carga |
| 15659 | Desenv | To Test | 3,00 | 5,00 | - | Erro | Normal | [2.10.2] [PRESENÇA CARGA] [EDICAO] Ao editar pedido o sistema náo persiste dos dados de novos documentos. |
| 15664 | Desenv | Done | 3,00 | 3,00 | - | Erro | Normal | [2.10.3][Presença de Carga] Corrigir Contador de quantidade no footer ao consultar ou incluir presenca Carga |
| 15694 | Desenv | To Do | - | 0 | - | Erro | Normal | [TRUNK] [LIBERAR EMBARQUE] - Sistema não exibe ocorrência ao tentar liberar embarque de presenca |
| 15695 | Test | To Do | - | 0 | - | Erro | Normal | [TRUNK] [LIBERAR EMBARQUE] - Sistema não exibe ocorrência ao tentar liberar embarque de presenca |
| **Sprint #117 - Number of tasks:** 28 | | | | | | | | |
| **Status:** Closed - **Duration:** 16 calendar days / 11 business days | | | | | | | | |
| **Goal:** | | | | | | | | |
| 29847 | Desenv | Done | 4,00 | 5,00 | Dev 1 | Planejada | Muito Alta | [Interno] [Conclusão Serviço] Disparar evento Pendência Concluída |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 29868 | Desenv | Done | 2,00 | 3,00 | Dev 1 | Planejada | Muito Alta | [Recepção] [Externo] Indicador de Cabotagem |
| 29927 | Desenv | Done | 16,00 | 34,10 | Dev 4 | Planejada | Muito Alta | [Posicionamento] [Externo] Cancelamento (RN_POSIC_15) |
| 29942 | Desenv | Done | 2,00 | 2,00 | Dev 1 | Planejada | Muito Alta | [Siscomex] [Escala] Importação do arquivo Siscomex está permitindo fazer associação de consignatário com cliente inativo na tabela de cliente |
| 29947 | Desenv | Done | 8,00 | 7,75 | Dev 4 | Planejada | Muito Alta | [Posicionamento] [Interno & Externo] Implementar dependência entre as pendências (RN_POSIC_09) |
| 29860 | Desenv | Done | 4,00 | 5,75 | Dev 8 | Melhoria | Alta | [Recepção] [Externo] Flags IMO, Reefer e OOG na adição de cargas |
| 27780 | Integracao | Done | 16,00 | 15,50 | Analist 3 | Planejada | Média | Integração Navis2PTVV Booking |
| 29945 | Desenv | Done | 8,00 | 15,65 | Dev 1 | Melhoria | Média | [Recepção] [Externo] [back-end] [front-end] [Pendência Carga Perigosa] Fix Carga Perigosa |
| 29985 | Integracao | Done | - | 7,50 | Analist 3 | Melhoria | Normal | Revisão dos Plug-ins (Conlusão Retirada e Conclusão Recepção) |
| 30139 | Integracao | Done | - | 24,35 | Analist 3 | Melhoria | Normal | Ambientes TST e HOMOL |
| 29745 | Desenv | Done | - | 0,25 | Dev 3 | Melhoria | Baixa | [Front-end][Layout] Mudar layout da sidebar de criação de serviço para utilizar botões ao invés de drop-downs |
| 30052 | Desenv | Done | - | 1,00 | Dev 3 | Melhoria | Baixa | [Front-end] Fixing de várias tarefas pendentes |
| 29918 | Desenv | To Test | 4,00 | 4,65 | Dev 1 | Planejada | Muito Alta | [Interno] [Cliente] [Front-end] [Back-end] Alterar atributo Carga.numeroONU para String |
| 29934 | Integracao | To Test | - | 0,00 | - | Planejada | Muito Alta | [Posicionamento] PTVV2Billing Posicionamento |
| 29935 | Integracao | To Test | - | 0,00 | - | Planejada | Muito Alta | [Posicionamento] Navis2PTVV Posicionamento |
| 29943 | Integracao | To Test | - | 0,00 | - | Planejada | Muito Alta | [Posicionamento] [Externo] Implementar Conclusão Posicionamento |
| 29933 | Integracao | To Test | 4,00 | 3,75 | Analist 3 | Planejada | Muito Alta | [Posicionamento] PTVV2Navis Posicionamento - Envio de ICU |
| 29941 | Desenv | To Test | 4,00 | 7,75 | Dev 1 | Planejada | Muito Alta | [Siscomex] [Atualização Escala] Importação do arquivo Siscomex não atualizou campos do CE mercante |
| 30026 | Integracao | To Test | - | 7,00 | Dev 3 | Não Planejada | Muito Alta | [LIQUIBASE] [UPDATE] App Event Action Config |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 30061 | Integracao | To Test | 3,00 | 3,00 | Dev 3 | Não Planejada | Muito Alta | [PROJETO EXTERNO/INTERNO] [AppEventPublisher] App Event Action Config |
| 30069 | Integracao | To Test | - | 3,00 | Dev 3 | Erro | Muito Alta | [Consumo Navis] Consumo Navis PTVV |
| 29929 | Desenv | To do | 8,00 | 11,00 | Dev 6 | Planejada | Muito Alta | Refactory TipoServico / SubtipoServico |
| 29930 | Desenv | To Review | 16,00 | 19,75 | Dev 6 | Planejada | Muito Alta | [Posicionamento] [Externo] Implementar Agendamento (RN_POSIC_10) |
| 29932 | Desenv | To Review | 16,00 | 28,00 | Dev 7 | Planejada | Muito Alta | [Posicionamento] [Externo] PDF Termo (RN_POSIC_11) |
| 29957 | Reuniao | Doing | - | 29,25 | - | - | Normal | Daily Meeting 17 |
| 29996 | Test | Doing | - | 18,25 | Analist 3 | Não Planejada | Normal | Testes de Integração |
| 29931 | Reuniao | To Do | - | 19,70 | - | Planejada | Normal | Pre-Planning da Sprint #117 |
| 29949 | Reuniao | To Do | - | 20,00 | - | Planejada | Normal | Planning da sprint #117 |
| **Sprint #118 - Number of tasks:** 30 | | | | | | | | |
| **Status:** Closed - **Duration:** 16 calendar days / 11 business days | | | | | | | | |
| **Goal:** | | | | | | | | |
| 29868 | Desenv | Done | 2,00 | 3,00 | Dev 1 | Planejada | Muito Alta | [Recepção] [Externo] Indicador de Cabotagem |
| 30130 | Desenv | Done | - | 2,50 | Dev 6 | Melhoria | Muito Alta | [Interno / Externo] Alterações no modelo para atender o retorno das requisições |
| 30035 | Desenv | Done | 9,00 | 20,50 | Dev 4 | Planejada | Alta | [Interno e Externo] Alterar estrutura Carga e criar novas entidades RetornoRequisicaoCCT, RequisicaoCCT e EventoGate |
| 30040 | Desenv | Done | 5,00 | 4,50 | Dev 4 | Planejada | Alta | [Interno] [back-end] Implementar consumidor para processar evento LIBERACAO_EMBARQUE |
| 30042 | Desenv | Done | 13,00 | 23,25 | Dev 8 | Planejada | Alta | [Interno] [front-end/back-end] [framework de busca] Tela de busca de escala |
| 30044 | Desenv | Done | 9,00 | 11,00 | Dev 4 | Planejada | Alta | [Interno] [back-end] Serviço para recuperar estrutura Armador/Carga por escalaId |

**Continued on next page**

| Continuation of Table G.1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Id** | **Type** | **Status** | **E.E.** | **R.E.** | **A.T.** | **Category** | **Priority** | **Title** |
| 30045 | Desenv | Done | 13,00 | 13,70 | Dev 1 | Planejada | Alta | [Interno] [back-end] Serviço de Entrega de Contêineres |
| 30046 | Desenv | Done | 13,00 | 7,25 | Dev 1 | Planejada | Alta | [Interno] [back-end] Serviço de Recepção de Contêineres |
| 30048 | Desenv | Done | 17,00 | 15,50 | Dev 7 | Planejada | Alta | [Interno] [front-end] Criar estrutura da tela de Entrega/Recepção Carga Contêiner |
| 30049 | Desenv | Done | 5,00 | 4,75 | Dev 7 | Planejada | Alta | "[Interno] [front-end] Implementar botões Entrega e Entregar Escala |
| 30051 | Desenv | Done | 9,00 | 15,75 | Dev 4 | Planejada | Alta | [front-end] Botão e modal de troca de Armador |
| 29936 | Desenv | Done | 2,00 | 2,75 | Dev 8 | Não Planejada | Normal | [Externo] [Pendência OOG] [Backend] Enviar evento ao resolver/reabrir pendência |
| 30022 | Reuniao | Done | - | 35,25 | - | Planejada | Normal | Pre-Planning da Sprint #118 |
| 30031 | Reuniao | Done | - | 42,50 | - | Planejada | Normal | Planning da Sprint #118 |
| 30066 | Reuniao | Done | - | 32,00 | - | - | Normal | Daily Meeting #118 |
| 30053 | Arquitetura | To Test | 8,00 | 5,50 | Dev 6 | Planejada | Muito Alta | [IPUE] Fazer um fork do projeto IPUE e criar projeto novoportaltvv-integracoes-portal-unico |
| 30077 | Integracao | To Test | 4,00 | 1,00 | Analist 3 | Planejada | Alta | [NAVIS2PTVV] [LIBERAR EMBARQUE] [TELA FIEL] Alteração do plugin |
| 30054 | Desenv | To Test | 8,00 | 6,50 | Dev 6 | Planejada | Alta | [IPUE] Refactoring no algoritmo de decisão de fluxo |
| 30050 | Desenv | To Review | 5,00 | 3,00 | Dev 7 | Planejada | Alta | "[Interno] [front-end] Botão Recepção (Cargas) |
| 30056 | Desenv | To Review | 8,00 | 3,00 | Dev 6 | Planejada | Alta | [IPUE] Refactoring no serviço Recepção por NFe |
| 30059 | Desenv | To Review | 4,00 | 3,25 | Dev 6 | Planejada | Alta | [IPUE] Refactoring no serviço Recepção por Contêiner (Processo manual) |
| 30038 | Desenv | To Review | 3,00 | 2,00 | Dev 4 | Planejada | Alta | [Externo] [back-end] Alterar fluxo criação Recepção |
| 30039 | Desenv | To Review | 5,00 | 5,90 | Dev 1 | Planejada | Alta | [Interno] [back-end] Alterar fluxo importação arquivo SISCOMEX |
| 30047 | Desenv | To Review | 7,00 | 3,50 | Dev 4 | Planejada | Alta | [back-end] Serviço de Troca de Armador por Contêineres |
| 30055 | Desenv | To Review | 8,00 | 6,00 | Dev 6 | Planejada | Alta | [IPUE] Incluir/atualizar registros de retorno dos serviços do Portal Único |
| Continued on next page | | | | | | | | |

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 30057 | Desenv | To Review | 8,00 | 4,00 | Dev 6 | Planejada | Alta | [IPUE] Refactoring no serviço Recepção por Contêiner (Processo automático) |
| 30058 | Desenv | To Review | 8,00 | 7,50 | Dev 6 | Planejada | Alta | [IPUE] Refactoring no serviço Entrega por Contêiner (Processo manual) |
| 29959 | Desenv | To Review | - | 4,25 | Dev 4 | Não Planejada | Normal | [Externo] [Pendência de Dados Documentais] Correções do codereview do branch CHRISTIAN-S15-T29587 |
| 30034 | Desenv | To Review | 8,00 | 13,55 | Dev 1 | Não Planejada | Normal | [Recepção] [Externo] [back-end] [front-end] [Pendência Carga Perigosa] Fix Carga Perigosa |
| 30060 | Desenv | To Review | 4,00 | 5,00 | Dev 6 | Melhoria | Normal | [Externo] [Posicionamento] Implementar melhorias no agendamento (AJUSTES DE REVIEW) |
| 30162 | Reuniao | To Do | - | 2,50 | - | Planejada | Normal | [REUNIAO] - Reuniao tratamento de bugs de retirada / recepcao |

**Sprint #119 - Number of tasks:** 42

**Status:** Closed - **Duration:** 16 calendar days / 12 business days

**Goal:**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 30174 | Desenv | Done | 3,00 | 2,25 | Dev 1 | Planejada | Muito Alta | [NIC] [NPTVV] Fazer script de insert na tabela PresencaCargaImportacaoCodigoErro |
| 30194 | Desenv | Done | 7,00 | 8,50 | Dev 4 | Planejada | Muito Alta | [Recepção Contêiner Cheio] [Cliente] Ajustar fluxo de criação de pendências |
| 30195 | Desenv | Done | 5,00 | 2,25 | Dev 4 | Planejada | Muito Alta | [Recepção Contêiner Cheio] [Interno] Ajustar fluxo de resolução de pendências financeiras |
| 30202 | Desenv | Done | 13,00 | 17,75 | Dev 7 | Planejada | Muito Alta | [VagaExtra] [Projeto Interno] [Back-end] [Front-end] Criar tela de consulta de pendências de aprovação/reprovação de vagas-extras |
| 30203 | Desenv | Done | 21,00 | 22,50 | Dev 1 | Planejada | Muito Alta | [VagaExtra] [Projeto Interno] [Back-end] [Retirada] Ajustar fluxo de Resolver Pendência Financeira |
| 30204 | Desenv | Done | 5,00 | 12,85 | Dev 1 | Planejada | Muito Alta | [VagaExtra] [Projeto Interno] [Recepção] [Back-end] Ajustar fluxo de Resolver Pendência Financeira |
| 30205 | Desenv | Done | 9,00 | 20,00 | Dev 7 | Planejada | Muito Alta | [VagaExtra] [Projeto Interno] [Back-end] [front-end] [Recepção] Aprovar Pendência de Aprovação de Vaga-Extra |
| 30206 | Desenv | Done | 7,00 | 3,25 | Dev 7 | Planejada | Muito Alta | [VagaExtra] [Projeto Interno] [Back-end] [Retirada] Aprovar Pendência de Vaga-Extra |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 30207 | Desenv | Done | 9,00 | 12,00 | Dev 6 | Planejada | Muito Alta | [VagaExtra] [Projeto Interno] [Back-end] [front-end] [Recepção] [Retirada] Reprovar Pendência de Vaga-Extra |
| 30208 | Desenv | Done | 17,00 | 21,00 | Dev 4 | Planejada | Muito Alta | [VagaExtra] [Projeto externo] [Agendamento] [Back-end] [Front-end] [Retirada] Alterar agendamento para permitir vaga extra |
| 30209 | Desenv | Done | 13,00 | 18,00 | Dev 4 | Planejada | Muito Alta | [VagaExtra] [Agendamento] [Recepção] [Back-end] Alterar agendamento para permitir vaga extra |
| 30340 | Desenv | Done | - | 0,25 | Dev 4 | Não Planejada | Muito Alta | [Agendamento Vaga Extra] [Projeto Interno] [front-end]Tela de agendamento quebrou após reprovar vaga extra |
| 30188 | Desenv | Done | 4,00 | 3,50 | Dev 4 | Planejada | Alta | [Cancelar CargaServico] [Projeto Externo] [back-end] Incluir acao de CANCELAR_CARGA_SERVICO na api que retorna acoesPermitidas |
| 30173 | Desenv | Done | 5,00 | 5,50 | Dev 1 | Planejada | Alta | [NIC] [NPTVV] [Interno] [Cliente] Criar tabelas PresencaCargaImportacaoErro e PresencaCargaImportacaoCodigoErro |
| 30180 | Desenv | Done | 17,00 | 15,50 | Dev 6 | Planejada | Alta | [Cancelar CargaServico] [back-end] [Recepcao e Posicionamento] Implementar cancelamento de cargas |
| 30183 | Desenv | Done | 7,00 | 10,25 | Dev 7 | Planejada | Alta | [Cancelar CargaServico] [Projeto Interno] [back-end] [front-end] Não exibir cargas canceladas nas telas de pendência |
| 30187 | Desenv | Done | 17,00 | 3,75 | Dev 1 | Planejada | Alta | [Cancelar CargaServico] [Projeto Externo] [back-end] Implementar metodo cancelarResolverPendencias ao cancelar cargaServico |
| 30189 | Desenv | Done | 2,00 | 1,00 | Dev 6 | Planejada | Alta | [Cancelar CargaServico] [front-end] [Recepção] [Posicionamento] Ajustar front-end para o cancelamento de cargas |
| 30199 | Desenv | Done | 3,00 | 2,25 | Dev 4 | Planejada | Alta | [Projeto Interno e Externo] Ajustar estrutura Agendamento/GradeHoraria (2h) |
| 30317 | Desenv | Done | - | 8,50 | Dev 6 | Erro | Alta | [Erro] Corrigir fluxo de confirmação de agendamento para serviços de Retirada |
| 30331 | Desenv | Done | - | 2,75 | Dev 4 | Não Planejada | Alta | [Interno] [back-end] [ERRO] Consumidor de pagamento de recepção não verifica se existem pendencias vaga extra ativas e reabre pendencia de agendamento |
| 30171 | Reuniao | Done | - | 48,55 | - | Planejada | Normal | Planning da sprint #119 |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 30198 | Reuniao | Done | - | 26,00 | - | Planejada | Normal | Pré-Planning da sprint #119 |
| 30250 | Reuniao | Done | - | 29,00 | - | - | Normal | Daily Meeting #119 |
| 29954 | Integracao | To Test | 2,00 | 1,50 | Dev 3 | Planejada | Muito Alta | [LIQUIBASE][POSICIONAMENTO] - Envio de ICU |
| 30230 | Integracao | To Test | 2,00 | 1,75 | Analist 3 | Não Desenvolvida | Alta | [N4] Alterar o envio das mensagens de gate-in para enviar também para a nova fila do IPUE |
| 29956 | Integracao | To Test | 1,00 | 0,50 | Dev 3 | Erro | Média | [LIQUIBASE] Erro ao enviar email de perda de vaga na grade |
| 30001 | Integracao | To Test | 1,00 | 0,50 | Dev 3 | Erro | Média | [LIQUIBASE] [EMAIL] - Pendencia reaberta |
| 30102 | Integracao | To Test | 1,00 | 1,00 | Dev 3 | Erro | Média | [LIQUIBASE] [DADOS TRANSPORTE] Update Appointment |
| 30114 | Integracao | To Test | 3,00 | 1,50 | Dev 3 | Erro | Média | [LIQUIBASE] [RECEPÇÃO CHEIO] Criação de Pre-Advise |
| 30299 | Integracao | To Test | 8,00 | 9,30 | Analist 3 | Não Planejada | Normal | [Booking] [Porto Descarga] Alteração dos mapas e configuração no N4 |
| 30305 | Desenv | To do | - | 6,50 | Dev 7 | Erro | Alta | [Interno] [Entrega Carga Contêiner] Alterar armador não valida statusCCT |
| 30196 | Desenv | To do | 9,00 | 6,00 | Dev 4 | Planejada | Normal | [Externo] [Recepção Contêiner Cheio] Ajustar fluxo de agendamento de carga |
| 30172 | Desenv | To Review | 5,00 | 4,50 | Dev 6 | Planejada | Muito Alta | [NIC] [NPTVV] Criar projeto no git da owse |
| 30175 | Desenv | To Review | 13,00 | 8,75 | Dev 6 | Planejada | Muito Alta | [NIC] [integracao-presencaCargaImportacao] Fazer persist na entidade PresencaCargaImportacaoNPTVV |
| 30191 | Desenv | To Review | 7,00 | 3,00 | Dev 6 | Planejada | Muito Alta | [NIC] [integracao-presencaCargaImportacao] processamento de retorno sucesso |
| 30192 | Desenv | To Review | 9,00 | 4,75 | Dev 6 | Planejada | Muito Alta | [NIC] [integracao-presencaCargaImportacao] processamento de retorno erro |
| 30076 | Integracao | To Review | 6,00 | 6,00 | Dev 3 | Planejada | Alta | [CONCLUSÃO SERVIÇOS] Retiradas e Recepção + mapeamento + liquibase |
| 30184 | Integracao | To Review | 2,00 | 1,00 | Analist 3 | Planejada | Alta | [RECEPÇÃO CHEIO] [PTVV2NAVIS] Mensagem de ICU |
| 30240 | Integracao | To Review | 4,00 | 3,00 | Analist 3 | Planejada | Alta | [RECEPÇÃO CHEIO] [PTVV2BILLING] Mensagem de Monitoramento Reefer |
| 30260 | Integracao | To Review | 3,00 | 2,50 | Analist 3 | Planejada | Alta | [VAGA EXTRA] [E-MAIL] Enviar e-mail Planejamento |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 30310 | Integracao | To Review | 1,00 | 1,25 | Analist 3 | Não Planejada | Normal | [HML/TST] [APP EVENT ACTION CONFIG] [FINANCEIRA CANCELADA] Alteração da mensagem de cancelamento para o Billing |
| **Sprint #120 - Number of tasks:** 89<br>**Status:** Open - **Duration:** 23 calendar days / 15 business days<br>**Goal:** | | | | | | | | |
| 30350 | Desenv | Done | 8,00 | 11,75 | Dev 6 | Não Desenvolvida | Imediata | [Cabotagem] [front-end] Permitir Criação de Serviço Retirada Contêiner Cabotagem como Agente de Carga |
| 30218 | Desenv | Done | 12,00 | 5,75 | Dev 4 | Não Desenvolvida | Urgente | [AcessoConsultaServico] Ajustar DAO para buscar injetar filtro de empresa requisitante |
| 30219 | Desenv | Done | 12,00 | 6,00 | Dev 4 | Não Desenvolvida | Urgente | [AcessoConsultaServicoCliente] Ajustar DAO para buscar injetar filtro de empresa consignatário |
| 30241 | Desenv | Done | 40,00 | 26,00 | Dev 7 | Não Desenvolvida | Urgente | [Acesso] Manter Usuários |
| 30030 | Desenv | Done | 13,00 | 12,90 | Dev 1 | Planejada | Muito Alta | [Arquivo siscomex] Atualização de escala siscomex não funciona corretamente |
| 30033 | Desenv | Done | 12,00 | 14,50 | Dev 6 | Planejada | Muito Alta | [Recepção e Posicionamento] [Externo] Finalizar o cancelamento de cargas |
| 30168 | Desenv | Done | 8,00 | 1,25 | Dev 4 | Erro | Alta | [Cancelar Serviço] [Posicionamento] [Posicionamento - Tratamento Pendência] [RN_POSIC_08] - Cancelamento de Posicionamento não esta funcionando |
| 30105 | Desenv | Done | 6,00 | 10,00 | Dev 4 | Erro | Alta | [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_12] - Sistema não resolve pendencia agendamento apos mensagem do billing de antecipacao de gate pago |
| 30266 | Integracao | Done | - | 2,00 | Dev 3 | Planejada | Alta | [LIQUIBASE] [VAGA EXTRA] [E-MAIL] Enviar e-mail Planejamento |
| 30332 | Desenv | Done | 4,00 | 1,00 | Dev 4 | Erro | Alta | [Projeto Cliente] [Recepção] [Pendencia Carga Perigosa] Botão Resolver Pendência não aparece |
| 30397 | Desenv | Done | - | 5,50 | Dev 4 | Não Planejada | Alta | [Posicionamento] [Projeto Interno] [back-end] Consumidor de pagamento não resolve pendencia de reagendamento |
| 30232 | Desenv | Done | 8,00 | 7,00 | Dev 7 | Não Desenvolvida | Média | [BoletimPesagem] Criar estrutura (entities, dtos, mapper, script liquibase, controller, repository) |
| 30233 | Desenv | Done | 16,00 | 17,50 | Dev 7 | Não Desenvolvida | Média | [BoletimPesagem] Tela de Busca (framework de busca, sem agrupamento, com paginação) |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 30342 | Reuniao | Done | - | 18,00 | - | Planejada | Normal | Reunião de planning e planning da Sprint #120 |
| 30394 | Reuniao | Done | - | 1,00 | - | - | Normal | Integração Ptvv2Billing |
| 30413 | Desenv | Done | 3,00 | 18,25 | Dev 9 | Não Planejada | Normal | [Retirada Cabotagem - Tratamento Pendência] [RN_CABOT_13] Correção de mascara na tela do fiel |
| 30464 | Desenv | Done | 3,00 | 2,25 | Dev 9 | Não Planejada | Normal | [Cadastro de Escala] - Colocar máscara de inteiro nos campos Abertura de Gate e Limite de Recepção Cheio |
| 30473 | Reuniao | Done | 1,00 | 2,50 | Analist 3 | - | Normal | Review Módulo de Integração NPTVV |
| 30421 | Desenv | Done | - | 3,60 | Dev 4 | Não Planejada | Normal | [Posicionamento - Tratamento Pendência] [RN_RECCONTCH_10] - Cancelamento de posicionamento não devolve as grades (capacidade) corretamente |
| 30483 | Reuniao | Done | - | 1,50 | - | - | Normal | Status Integração |
| 30375 | Integracao | To Test | - | 6,00 | Dev 3 | Erro | Imediata | [LIQUIBASE][Recepção cheio - Criação Serviço] [RN_RECCONTCH_09] - Sistema deve enviar primeiro o preadvise para o N4, depois as demais integrações |
| 30243 | Desenv | To Test | 8,00 | 5,25 | Dev 4 | Não Desenvolvida | Muito Alta | [ReagendamentoPosicionamento] Alterar a criação de serviço para criar a multa (indicação multa) |
| 30024 | Desenv | To Test | 6,00 | 0,75 | Dev 4 | Erro | Alta | [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_12] - Erro ao tentar resolver a pendencia financeira (scanner) |
| 30025 | Desenv | To Test | 5,00 | 1,00 | Dev 4 | Erro | Alta | [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_12] - Erro ao tentar resolver a pendencia financeira (VGM) |
| 30185 | Integracao | To Test | - | 25,00 | Dev 3 | Planejada | Alta | [LIQUIBASE] [RECEPÇÃO CHEIO] Mensagem de ICU |
| 30322 | Integracao | To Test | - | 1,00 | Dev 3 | Planejada | Alta | [LIQUIBASE] [RECEPÇÃO CHEIO] [PTVV2BILLING] Mensagem de Monitoramento Reefer |
| 30311 | Integracao | To Test | - | 3,00 | Dev 3 | Não Planejada | Normal | [HML/TST] [LIQUIBASE] [FINANCEIRA CANCELADA] Alteração da mensagem de cancelamento para o Billing |
| 30435 | Integracao | To Test | - | 2,00 | Dev 3 | Erro | Normal | [LIQUIBASE] [HML/TST/DSV] [Constraint][BOLETIMPESAGEM] Mais de 30 posicoes |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 30373 | Integracao | To Test | - | 1,00 | Dev 3 | Erro | Normal | [LIQUIBASE] [Retirada Cabotagem - Tratamento Pendência] [RN_CABOT_09] - Sistema deve enviar e-mail caso a solicitação seja reagendada |
| 30401 | Integracao | To Test | - | 4,00 | Dev 3 | Erro | Normal | [LIQUIBASE] [Posicionamento - Criação Serviço] [RN_POSIC_09] - Sistema não envia integração pro billing na criação da solicitação. |
| 30432 | Integracao | To Test | - | 2,50 | Dev 3 | Erro | Normal | [LIQUIBASE] [HML/TST] [PTVV2Billing] Alteração dos indicadores de imo, reefer e oog para string |
| 30433 | Integracao | To Test | - | 2,00 | Dev 3 | Erro | Normal | [LIQUIBASE] [HML/TST] [PTVV2Billing][CABOTAGEM] Criação de pendência armazenagem adicional |
| 30438 | Integracao | To Test | - | 1,00 | Dev 3 | Erro | Normal | [LIQUIBASE][HML/TST][Posicionamento - Tratamento Pendência] [RN_POSIC_12_PTVV2Navis] - Sistema não envia mensagem de integração para o navis |
| 30482 | Integracao | To Test | - | 1,00 | - | Não Planejada | Normal | [LIQUIBASE][Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_07] - Sistema nao gera um ADD HOLD de PENDENCIA LIBERACAO_MEIO_AMBIENTE |
| 30485 | Integracao | To Test | - | 2,00 | Dev 3 | Erro | Normal | [Billing2PTVV] Autorização de execução |
| 30492 | Integracao | To Test | - | 1,50 | Dev 3 | Não Planejada | Normal | [LIQUIBASE] [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_12] - Incluir Excesso lateral esquerda no script de OOG |
| 30498 | Integracao | To Test | - | 3,50 | Dev 3 | Não Planejada | Normal | [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_12] - Resolução pendencia de OOG não envia mensagem de OOG para o N4 |
| 30224 | Desenv | To do | 8,00 | 13,15 | Dev 4 | Não Desenvolvida | Imediata | [Cabotagem/Importacao] [back-end] Implementar ajustes na busca de cargas |
| 27973 | Desenv | To do | 13,00 | 24,05 | Dev 1 | Planejada | Muito Alta | [Arquivo Siscomex] Popular Bloqueios nas Cargas |
| 30267 | Desenv | To do | 4,00 | 1,00 | Dev 5 | Erro | Alta | [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_19] - Sistema esta permitindo salvar a NFE sem dados obrigatórios |
| 30442 | Desenv | To do | 16,00 | 10,50 | Dev 7 | Não Planejada | Normal | [BoletimPesagem] [Erro de Planejamento] Adaptar tela de busca de boletins para perfis de Cliente e Despachante |
| 29953 | Desenv | Reviewing | 16,00 | 7,25 | Dev 5 | Erro | Média | [Retirada Cabotagem - Tratamento Pendência] [Cancelamento Serviço] Pendencia financeira criada com subtipo reagendamento não gera linha na tbl indicar multa apos cancelar solicitação |

**Continued on next page**

| Continuation of Table G.1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Id** | **Type** | **Status** | **E.E.** | **R.E.** | **A.T.** | **Category** | **Priority** | **Title** |
| 30215 | Desenv | To Review | 8,00 | 6,00 | Dev 6 | Não Desenvolvida | Imediata | [Recepcao] [front-end] Permitir Criação de Serviço Recepcao como Despachante |
| 30220 | Desenv | To Review | 8,00 | 8,25 | Dev 6 | Não Desenvolvida | Imediata | [Recepcao] [front-end] Permitir Criação de Serviço Recepcao como Transportador |
| 30223 | Desenv | To Review | 8,00 | 7,35 | Dev 6 | Não Desenvolvida | Imediata | [Importacao] [front-end] Permitir Criação de Serviço Retirada Contêiner Importação como Despachante |
| 30328 | Integracao | To Review | 6,00 | 6,75 | Analist 3 | Erro | Imediata | [Recepção cheio - Criação Serviço] [RN_RECCONTCH_09] - Sistema deve enviar primeiro o preadvise para o N4, depois as demais integrações |
| 30341 | Desenv | To Review | 4,00 | 5,00 | Dev 4 | Planejada | Imediata | [MóduloInterno] Ajustar acesso as funcionalidades para permitir apenas ADMIN |
| 30472 | Integracao | To Review | 1,00 | 1,00 | Analist 3 | Não Desenvolvida | Imediata | [PTVV2Billing] [RECEPÇÃO] Vaga extra |
| 30136 | Desenv | To Review | 24,00 | 28,75 | Dev 4 | Planejada | Muito Alta | [ReagendamentoPosicionamento] Reagendamento Serviço Posicionamento |
| 30242 | Desenv | To Review | 16,00 | 8,00 | Dev 4 | Não Desenvolvida | Muito Alta | [ReagendamentoPosicionamento] Aplicar multa ao reagendar |
| 30234 | Desenv | To Review | 16,00 | 18,40 | Dev 1 | Não Desenvolvida | Muito Alta | [BoletimPesagem] Complementar resultado tela de busca com informações Navis (a confirmar) |
| 30229 | Desenv | To Review | 6,00 | 1,00 | Dev 6 | Não Desenvolvida | Alta | [IPUE] Alterar o nome de todas as filas consumidas para não roubar mensagens do legado |
| 30361 | Desenv | To Review | 13,00 | 25,05 | Dev 1 | Planejada | Alta | [Cancelar CargaServico] [Projeto Externo] [back-end] Implementar metodo cancelarResolverPendencias ao cancelar cargaServico |
| 30478 | Desenv | To Review | 6,00 | 6,00 | Dev 4 | Erro | Alta | [Recepção cheio] [Projeto externo] [Finalizar Agendamento] - Ajustes ao finalizar agendamentos. |
| 29925 | Desenv | To Review | 6,00 | 5,25 | Dev 4 | Erro | Média | [Retirada Cabotagem - Tratamento Pendência] [RN_CABOT_13] - Alterar fluxo de confirmação de Agendamento para cancelar agendamentos que já venceram |
| 30376 | Desenv | To Review | 2,00 | 2,00 | Dev 6 | Não Desenvolvida | Média | [IPUE] Atualizar Certificado |
| 30422 | Integracao | To Review | 3,00 | 3,00 | Analist 3 | Não Desenvolvida | Normal | Migrar Peso Bruto Conteiner do Navis |
| 30429 | Integracao | To Review | 2,00 | 1,50 | Analist 3 | Erro | Normal | [HML/TST] [PTVV2Billing] Alteração dos indicadores de imo, reefer e oog para string |
| 30440 | Integracao | To Review | 1,00 | 1,00 | Analist 3 | Erro | Normal | [Posicionamento - Tratamento Pendência] [RN_POSIC_12_PTVV2Navis] - Sistema não envia mensagem de integração para o navis |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 30374 | Integracao | To Review | 1,00 | 4,00 | Analist 3 | Erro | Normal | [Posicionamento - Criação Serviço] [RN_POSIC_09] - Sistema não envia integração pro billing na criação da solicitação. |
| 30456 | Integracao | To Review | 10,00 | 14,00 | Analist 3 | Erro | Normal | [PTVV2Billing][RETIRADA CABOT/IMPORT] Alteração das mensagens de cobrança |
| 30491 | Integracao | To Review | 1,00 | 0,25 | Analist 3 | Não Planejada | Normal | [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_12] - Incluir Excesso lateral esquerda no script de OOG |
| 30502 | Desenv | To Review | 2,00 | 0,00 | Dev 5 | Melhoria | Normal | [Externo] Adicionar CNPJ e Endereço na página de empresas |
| 30479 | Desenv | To Review | 12,00 | 9,00 | Dev 5 | Planejada | Normal | [Externo] Alterações de layout |
| 30480 | Integracao | To Review | 0,50 | 1,00 | Analist 3 | Não Planejada | Normal | [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_07] - Sistema nao gera um ADD HOLD de PENDENCIA LIBERACAO_MEIO_AMBIENTE |
| 30235 | Desenv | Doing | 16,00 | 12,00 | Dev 7 | Não Desenvolvida | Muito Alta | [BoletimPesagem] Tela de Detalhes |
| 30236 | Desenv | Doing | 16,00 | 6,00 | Dev 7 | Não Desenvolvida | Muito Alta | [BoletimPesagem] PDF do Boletim Pesagem |
| 30244 | Integracao | Doing | 16,00 | 0,00 | Dev 3 | Planejada | Normal | [Booking] [Code Extension] - Disparo de evento após validar o booking |
| 30416 | Desenv | Doing | 6,00 | 0,00 | Dev 9 | Não Planejada | Normal | [Retirada Cabotagem - Tratamento Pendência] [RN_CABOT_13] - Apos a suspensão sistema não altera a solicitação para o status pendente |
| 30231 | Integracao | To do | - | 0,00 | Dev 3 | Não Desenvolvida | Muito Alta | [BoletimPesagem] Integrar dados SBP/NPTVV |
| 30186 | Integracao | To do | - | 0,00 | Dev 3 | Não Desenvolvida | Muito Alta | [BOOKING][NAVIS2PTVV] Consumir Booking |
| 30475 | Integracao | To do | - | 0,00 | - | Não Desenvolvida | Muito Alta | [LIQUIBASE] [PTVV2Billing] [RECEPÇÃO] Vaga extra |
| 30296 | Desenv | To do | 8,00 | 0,00 | - | Não Planejada | Alta | [Retirada Cabotagem - Tratamento Pendência] [RN_CABOT_13] - Sistema não envia integração para o billing corretamente ( reagendamento ) |
| 30460 | Integracao | To do | - | 0,00 | Dev 3 | Erro | Alta | [LIQUIBASE] [PTVV2Billing][RETIRADA CABOT/IMPORT] Alteração das mensagens de cobrança |
| 29557 | Integracao | To do | 16,00 | 0,00 | Dev 3 | Erro | Média | [Booking] Ajustes em Code Extensions |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 30238 | Desenv | To do | 24,00 | 0,00 | - | Não Desenvolvida | Média | [ConsultaServico] Opção de pesquisar por BL/Contêiner/Booking |
| 30439 | Integracao | To do | - | 0,00 | Dev 3 | Não Desenvolvida | Média | Consumo Peso Bruto Conteiner do Navis |
| 30360 | Reuniao | To do | - | 39,00 | - | - | Normal | Daily Meeting #120 |
| 30393 | Desenv | To do | 8,00 | 0,00 | - | Não Planejada | Normal | Criar script no liquibase do Novo PTVV com as configurações do Novo IPUE e PresencaCargaImportacao (NIC) |
| 30477 | Desenv | To do | - | 0,00 | - | Não Planejada | Normal | [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_11] - Sistema esta gerando antecipação de gate incorretamente |
| 30488 | Desenv | To do | - | 0,00 | - | Não Planejada | Normal | [Importação - Pendência] RN_SERVICOS_CANCELAMENTO - Solicitação com status cancelado não exibe suas cargas |
| 30501 | Desenv | To do | - | 0,00 | - | Não Planejada | Normal | [Recepção cheio - Criação Serviço] [RN_RECCONTCH_12] - Pendencia de scanner não esta sendo criada da recepção com booking |
| 30503 | Desenv | To do | - | 0,00 | - | Não Planejada | Normal | [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_12] - Pendencia de dados documentais só pode ser criada para pedidos tipo Longo curso |
| 30508 | Desenv | To do | - | 0,00 | - | Não Planejada | Normal | [Importação Siscomex] - Processo de importação do siscomex não popula a coluna CARGA_-CONSIGNATARIO.TIPO_CONHECIMENTO |
| 30486 | Desenv | To do | - | 0,00 | - | Não Planejada | Normal | [Retirada Importação - Criação Serviço] [RN_CABOT_04] - Não exibir para o usuário problemas de conectividade entre sistemas log-in |
| 30419 | Desenv | To do | - | 2,50 | - | Não Planejada | Baixa | [Siscomex] - Arquivo de produção nao é importado no ambiente de teste |
| 30227 | Desenv | To do | 8,00 | 0,00 | - | Não Desenvolvida | Média | [Cabotagem] [front-end] Permitir Criação de Serviço Retirada Contêiner Cabotagem como Agencia Marítima |
| 30225 | Desenv | To do | 8,00 | 0,00 | - | Não Desenvolvida | Média | [Cabotagem] [front-end] Permitir Criação de Serviço Retirada Contêiner Cabotagem como Armador |
| 30400 | Desenv | To do | - | 0,00 | - | Não Planejada | Média | [Externo] [Criação do Serviço de Retirada] Recuperar o consignatário após a consulta de cargas por CE |

**Continued on next page**

**Continuation of Table G.1**

| Id | Type | Status | E.E. | R.E. | A.T. | Category | Priority | Title |
|---|---|---|---|---|---|---|---|---|
| 30494 | Desenv | To do | - | 0,00 | - | Não Planejada | Normal | [Recepção cheio - Tratamento Pendência] [RN_RECCONTCH_12] Criação de evento Dados OOG alterados |

**End of Table G.1**