

DISSERTAÇÃO DE MESTRADO

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM

CIÊNCIA DA COMPUTAÇÃO

**“ScrumOntoBDD: uma abordagem baseada
em Scrum, Ontologia e BDD para o
desenvolvimento ágil de software”**

ALUNO: Pedro Lopes de Souza
ORIENTADOR: Prof. Dr. Antonio Francisco do Prado

São Carlos
Agosto / 2018

CAIXA POSTAL 676
FONE / FAX: (16) 3351-8233
13565-905 - SÃO CARLOS - SP
BRASIL

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

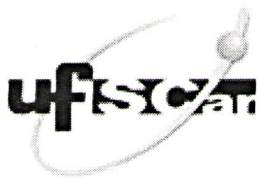
**ScrumOntoBDD: uma abordagem baseada em
Scrum, Ontologia e BDD para o desenvolvimento
ágil de software**

PEDRO LOPES DE SOUZA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Área de concentração: Engenharia de Software

Orientador: Prof. Dr. Antonio Francisco do Prado




UNIVERSIDADE FEDERAL DE SÃO CARLOS


Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

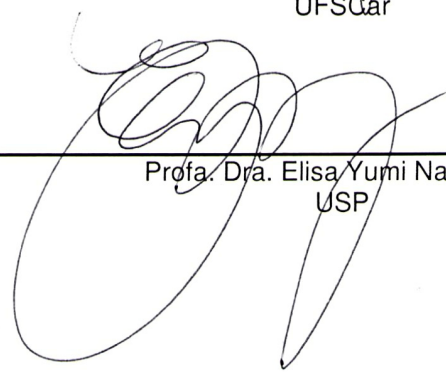
Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Pedro Lopes de Souza, realizada em 27/08/2018:



Prof. Dr. Antonio Francisco do Prado
UFSCar



Prof. Dr. Valter Vieira de Camargo
UFSCar



Profa. Dra. Elisa Yumi Nakagawa
USP

SÍNTESE

A maioria das universidades brasileiras empregam metodologias tradicionais de ensino-aprendizagem, baseadas em disciplinas e aulas expositivas. A Universidade Federal de São Carlos (UFSCar) não é uma exceção, mas alguns de seus cursos empregam metodologias ativas de aprendizagem, tais como Aprendizagem Baseada em Problemas (ABP). O Grupo de Computação Ubíqua (GCU) da UFSCar, criado em 2002, vem desenvolvendo projetos principalmente voltados para as áreas de Educação e Saúde e recentemente coordenou o desenvolvimento do projeto Software de Gestão Pedagógica e Acadêmica para Cursos Baseados em Metodologias Ativas de Aprendizagem (SGPA-CBMAA). O método ágil Scrum foi empregado no desenvolvimento do SGPA-CBMAA, tendo sido realizadas reuniões periódicas envolvendo desenvolvedores e Product Owners (POs) para o planejamento e análise das diferentes fases do desenvolvimento do SGPA-CBMAA. Foi bastante comum a necessidade de refazer cenários de comportamentos desse sistema, devido às ambiguidades presentes em especificações de requisitos, ou devido às interpretações equivocadas de estórias relatadas pelos POs. A definição de conjuntos de teste também foi incômoda, resultando em conjuntos de teste incompletos ou que não atendiam aos requisitos do sistema. Com base nessa experiência e para lidar com esses problemas, este trabalho propõe a abordagem ScrumOntoBDD, que combina Scrum, Ontologia e Behaviour-Driven Development (BDD), para o desenvolvimento ágil de software. Essa abordagem explora os conceitos e técnicas de SCRUM e BDD, focando nas fases de planejamento e análise do ciclo de vida do software, já que as ferramentas do BDD fornecem pouco apoio a essas fases, e a maioria dos problemas encontrados no desenvolvimento do SGPA-CBMAA foram nas mesmas. ScrumOntoBDD emprega ontologias a fim de eliminar as ambiguidades intrínsecas ao uso de uma linguagem natural como linguagem ubíqua do BDD.

Palavras-chave: Desenvolvimento Ágil de Software, Scrum, BDD, Ontologia, Sistema de Gerenciamento de Aprendizagem, Metodologia Ativa de Aprendizagem, ABP.

FEDERAL UNIVERSITY OF SÃO CARLOS

EXACT AND TECHNOLOGY SCIENCES CENTER
GRADUATE PROGRAMME ON COMPUTER SCIENCE

**ScrumOntoBDD: an approach based on Scrum,
Ontology and BDD for agile software development**

PEDRO LOPES DE SOUZA

ADVISOR: PROF. DR. ANTONIO FRANCISCO DO PRADO

FEDERAL UNIVERSITY OF SÃO CARLOS

EXACT AND TECHNOLOGY SCIENCES CENTER
GRADUATE PROGRAMME ON COMPUTER SCIENCE

**ScrumOntoBDD: an approach based on Scrum,
Ontology and BDD for agile software development**

PEDRO LOPES DE SOUZA

Dissertation presented to the Graduate Programme on Computer Science of the Federal University of São Carlos, as part of the requirements for obtaining a MSc's degree in Computer Science

Concentration area: Software Engineering

Advisor: Prof. Dr. Antonio Francisco do Prado

FEDERAL UNIVERSITY OF SÃO CARLOS

EXACT AND TECHNOLOGY SCIENCES CENTER
GRADUATE PROGRAMME ON COMPUTER SCIENCE

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

PEDRO LOPES DE SOUZA

Dissertation presented to the Graduate Programme on Computer Science of the Federal University of São Carlos, as part of the requirements for obtaining a MSc's degree in Computer Science

Concentration area: Software Engineering

Approved on August ____ 2018.

Graduation Committee:

Prof. Dr. Antonio Francisco do Prado

(Advisor – DC/CCET/UFSCar)

Prof. Dr. Valter Vieira de Camargo

(DC/CCET/UFSCar)

Prof. Dr. Elisa Yumi Nakagawa

(SSC/ICMC/Universidade de São Paulo)

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

*I devote this work to my parents who's always believed
in me, even at times where I didn't believe in myself.*

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

“Learning is the only thing the mind never exhausts, never fears, and never regrets.”

- Leonardo di ser Piero da Vinci

ACKNOWLEDGEMENTS

This work would not have been possible without the help of important people around me. I will start by thanking my parents Wanderley e Inês for always trusting me, and supporting me during this entire journey. I am grateful to my sister Izabel and to my brother Adriano for all their love and care through these years of physical distance.

My sincere appreciation to Prof. Dr. Antonio Francisco do Prado for supervising me, encouraging my research, and for his patience, motivation, and enthusiasm through all the development of this MSc's project.

My gratitude to Prof. Dr. Luís Ferreira Pires for his ideas and advices to improve this work, and for his intense participation in the publications resulting from this research.

I would like to thanks two people for advising me in essential matters for this project development: Prof. Sissi Marília dos Santos Forghieri Pereira, an expert on the application domain addressed in this work; and Prof. Helen de Freitas Santos, an expert on the software system addressed in this work.

I am grateful to all my professors of the Graduate Programme on Computer Science (PPG-CC) of Federal University of São Carlos (UFSCar) for improving my knowledge with their lectures. Special thanks to Prof. Dr. Vânia Paula de Almeida Neris for helping me to define the evaluation process of the approach proposed in this MSc's dissertation.

I am honoured to have Prof. Dr. Antonio Francisco Prado and Prof. Dr. Valter Vieira de Camargo from Computing Department (DC) of UFSCar, and Prof. Dr. Elisa

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

Yumi Nakagawa from Institute of Mathematical and Computer Sciences (ICMC) of University of São Paulo (USP) on the defence committee of my MSc's dissertation.

I thank my fellow lab mates in UFSCar for the stimulating discussions, for working together mainly before deadlines, and for all the fun we have had in the last two years.

Last but not least, this study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001- and I would like to thanks this agency for sponsoring this research.

ABSTRACT

Most Brazilian universities employ traditional teaching-learning methodologies based on lectures classes. The Federal University of São Carlos (UFSCar) is not an exception, but some of its programmes employ active learning methodologies, such as Problem Based Learning (PBL). The Ubiquitous Computing Group (UCG) of UFSCar, which was established in 2002, has been developing projects focused mainly on the Education and Health areas, and recently coordinated the development of the project Educational and Academic Management System for Courses Based on Active Learning Methodologies (EAMS-CBALM). The Scrum agile method was employed in the EAMS-CBALM development, with periodic meetings involving developers and Product Owners (POs) for planning and analyzing the different EAMS-CBALM development phases. It was quite often necessary to redefine some system behaviour scenarios, due to ambiguities present in requirement specifications, or due to misinterpretations of stories reported by POs. The definition of test suites was also cumbersome, resulting in test suites that were incomplete or did not at all comply with the system requirements. Based on this experience and to deal with these problems, this work proposes the ScrumOntoBDD approach, which combines Scrum, Ontology and Behaviour-Driven Development (BDD), for agile software development. This approach explores the concepts and techniques of SCRUM and BDD, focusing on the planning and analysis phases of software life cycle, since the BDD tools provide little support to these phases, and most of the problems found in the EAMS-CBALM development were in those phases. ScrumOntoBDD employs ontologies in order to eliminate ambiguities intrinsic to the use of a natural language as a BDD ubiquitous language.

Keywords: Agile Software Development, Scrum, BDD, Ontology, Learning Management System, Active Learning Methodology, PBL.

FIGURES LIST

Figure 1 – Scrum roles, events, and artefacts	7
Figure 2 – A the top chart, Google Trends research. At the botton chart, Scopus publications database (from(Dingsoyr, 2016)).....	9
Figure 3 – The BDD process in SADT.....	12
Figure 4 – BDD templates for user stories and scenarios	14
Figure 5 – Points for connecting BDD to Scrum	16
Figure 6 – Ontology classification.....	17
Figure 7 – Key concepts of HERO ontology	19
Figure 8 – Excerpt of HERO ontology in OWL.....	21
Figure 9 – Protégé 5.2.0 screenshot of the HERO ontology.....	25
Figure 10 – UFSCar Medicine Programme curriculum.....	26
Figure 11 – Movements of the constructivist spiral.....	27
Figure 12 – EAMS-CBALM academic module.....	30
Figure 13 – EAMS-CBALM pedagogical module.....	30
Figure 14 – Architectural pattern of EAMS-CBALM implementation	31
Figure 15 – ScrumOntoBDD overview	35
Figure 16 – HERO key concepts with CNPq major education areas.....	38
Figure 17 – UFSCar ontology in OWLViz.....	39
Figure 18 – UFSCar Medicine Programme ontology.....	40
Figure 19 – Constructivist spiral steps and educational activities of UFSCar Medicine Programme ontology	40
Figure 20 – Evaluation Process ontology root classes (left-side) and <i>Meeting</i> class relations (right-side).....	43
Figure 21 – Evaluation Process ontology with PATLP	43

Figure 22 – Results of Evaluation Process ontology processing: FaCT++ (top), HermiT (middle) and Pellet (bottom).....	44
Figure 23 – Use case diagram for Instrument Types Registration requirement.....	46
Figure 24 – System feature set for Instrument Types Registration requirement.....	47
Figure 25 – OWL ontology for the BDD scenario template.....	48
Figure 26 – JBehave User Story and Scenario templates.....	50
Figure 27 – EFSM model representing PATLP instrument type registration	50
Figure 28 – User Story and Scenario ontology with PATLP instrument type registration.....	51
Figure 29 – User Story and Scenario ontology transcript to JBehave textual story file	53
Figure 30 – Excerpt of the acceptance tests generated for the PATLP scenario	55
Figure 31 – Excerpt of the achieved implementation for the Instrument Types Registration requirement	56
Figure 32 – Evaluation Process ontology with extended PATLP.....	57
Figure 33 – Action Research cycle.....	60
Figure 34 – AR cycles of experimental analysis.....	64
Figure 35 – Activities in the Action Planning phases of the AR cycles	66
Figure 36 – Excerpt of UML use case diagram for Create People Performance Evaluation Format requirement	67
Figure 37-A – Excerpt of the Add People Performance Evaluation Format feature...	68
Figure 37-B – Excerpt of the Add People Performance Evaluation Format feature...	69
Figure 38 – Excerpt of Extended Evaluation Process ontology focusing on People Performance Evaluation Format.....	70
Figure 39 – Excerpt of the PB item selected in Cycle 1.....	70
Figure 40 – Excerpt of the PB item selected in Cycle 2 focusing on the selection of the target group for the evaluation format scenario	71
Figure 41 – Screen prototype for the selected PB item	71
Figure 42 – Excerpt of the acceptance tests generated in Cycle 2	72
Figure 43 – Excerpt of the achieved implementation on Cycle 1.....	73
Figure 44 – Excerpt of the achieved implementation on Cycle 2.....	74

TABLES LIST

Table 1 – Supported BDD characteristics by the analyzed BDD tools.....	11
Table 2 – Supported programming languages by the analyzed BDD tools.....	11
Table 3 – Inputs, outputs, resources, roles and tasks of CDAO activity.....	37
Table 4 – Evaluation instruments types.....	42
Table 5 – Inputs, outputs, resources, roles and tasks of EAR activity.....	45
Table 6 – Inputs, outputs, resources, roles and tasks of BPB activity.....	49
Table 7 – Inputs, outputs, resources, roles and tasks of DSB activity.....	52
Table 8 – Inputs, outputs, resources, roles and tasks of ES activity.....	54
Table 9 – Inputs, outputs, resources, roles and tasks of SRR activity.....	56
Table – 10 Main results of the Evaluating phase of Cycle 1.....	88
Table – 11 Main results of the Evaluating phase of Cycle 2.....	89

ACRONYMS LIST

ADD – Application Domain Documents

AR – Action Research

AS – Application Scenarios

ATDD – Acceptance Test Driven Development

BDD – Behaviour-Driven Development

BDDT – BDD Tools

BDDUSS – BDD User Stories and Scenarios

BPB – Building Product Backlog

BPMN – Business Process Model Notation

BPMT – Business Process Modeling Techniques

CBALM – Courses Based on Active Learning Methodologies

CDAO – Creating Domain Application Ontologies

DAML+OIL – DARPA Agent Markup Language + Ontology Interchange Language

DE – Domain Expert

DL – Description Logic

DMed – Medicine Department

DR – Design Research

DRO – Domain Reference Ontologies

DSB – Defining Sprint Backlog

DT – Development Team

EAMS-CBALM – Educational and Academic Management System for Courses
Based on Active Learning Methodologies

EAR – Extracting Application Requirements

EFSM – Extended Finite State Machine

EPM – Paulista Medical School

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

ERP – Electronic Reflective Portfolio

ES – Executing Sprint

EUEA – Education Unit of Elective Activities

EUPP – Education Unit of Professional Practice

EUSPP – Education Unit of Simulation of Professional Practice

Famema – Faculty of Medicine of Marília

FE – Free Edition

F-logic – Frame Logic

FS – Feature Sets

FSP – Faculty of Public Health

H₁ – Hypothesis 1

H₂ – Hypothesis 2

HERO – Higher Education Reference Ontology

ICT – Information and Communication Technologies

IFSP – Federal Institute of Education, Science and Technology of São Paulo

IS – Information Systems

KIF – Knowledge Interchange Format

KMI – Knowledge Media Institute

LMS – Learning Management System

ME – Maestro Edition

MIND – Maryland Information and Network Laboratory

MVC – Model–View–Controller

OCML – Operational Conceptual Modelling Language

OE – Ontology Engineer

OKBC – Open Knowledge Base Connectivity

OL – Ontology Languages

OSEPP – Objective and Structured Evaluation of Professional Performance

OT – Ontology Tools

OWL – Web Ontology Language

PATLP – Performance Assessment of the Teaching-Learning Process

PB – Product Backlog

PBE – Problems Based Exercise

PBL – Problem Based Learning

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

PI – Product Increment

PL – Programming Languages

PO – Product Owner

PPG-CC – Graduate Programme on Computer Science

PT – Progress Test

PTo – Programming Tools

RDF – Resource Description Framework

RDFS – RDF Schema

RP – Reflective Portfolio

RQ₁ – Research Question 1

RQ₂ – Research Question 2

SADT – Structured Analysis and Design Technique

SB – Sprint Backlog

SBe – System Behaviours

ScrumOntoBDD – Approach based on Scrum, Ontology and BDD for agile software development

SE – Software Engineer

SEd – Standard Edition

SHOE – Simple HTML Ontology Extensions

SLH – Sírío-Libanês Hospital

SM – Scrum Master

SP – Shippable Product

SPr – Screen Prototypes

SRA – Sprint Records and Annotations

SRR – Sprint Review and Retrospective

ST – Scrum Team

SWAP – Semantic Web and Agents Project

SWOOP – Semantic Web Ontology Overview and Perusal

SWRL – Semantic Web Rule Language

TDD – Test Driven Development

TRI – Teaching and Research Institute

UCG – Ubiquitous Computing Group

UEM – State University of Maringá

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

UFMG – Federal University of Minas Gerais

UFSCar – Federal University of São Carlos

UI – User Interface

ULT – Ubiquitous Language Terminology

UML – Unified Modeling Language

UNICAMP – University of Campinas

UNIFESP – Federal University of São Paulo

USNL – User Stories in Natural Language

USP – University of São Paulo

W3C – World Wide Web Consortium

WE – Written Examination

WG – Working Group

XML – EXtensible Markup Language

CONTENTS

CHAPTER 1 - Introduction	1
1.1 Context	1
1.2 Motivations and Objectives	2
1.3 Structure of the Dissertation	3
CHAPTER 2 - THEORETICAL AND TECHNICAL BACKGROUND	5
2.1 Agile Software Development	5
2.1.1 Scrum	6
2.1.2 Behaviour-Driven Development (BDD)	10
2.1.2.1 Ubiquitous Language	12
2.1.2.2 BDD Templates	13
2.1.2.3 Acceptance Tests	14
2.1.2.4 Implementation	15
2.1.3 Combining Scrum and BDD	15
2.2 Ontologies	16
2.2.1 Ontology Classification	17
2.2.2 Ontology Languages	19
2.2.3 Ontology Tools	21
2.3 Active Learning Methodologies	25
2.3.1 UFSCar Medicine Programme	26
2.4 Learning Management Systems	27
2.4.1 Educational and Academic Management System for Courses Based on Active Learning Methodologies (EAMS-CBALM)	28
2.4.1.1 EAMS-CBALM Functional Architecture	29

2.4.1.2 EAMS-CBALM Implementation	31
2.5 Final Considerations.....	31
CHAPTER 3 - SCRUMONTOBDD.....	34
3.1 Overview.....	34
3.2 Creating Domain Application Ontologies (CDAO)	37
3.2.1 Ubiquitous Language Terminology for EAMS-CBALM	38
3.2.1.1 Ubiquitous Language Terminology for Evaluation Management Module.....	41
3.2.2 Ontology Validation of EAMS-CBALM.....	44
3.3 Extracting Application Requirements (EAR).....	45
3.3.1 Requirements for EAMS-CBALM.....	46
3.4 Building Product Backlog (BPB).....	48
3.4.1 User Story and Scenario Ontology for EAMS-CBALM.....	49
3.5 Defining Sprint Backlog (DSB).....	51
3.5.1 Tasks for Implementing the Selected BDDUSS of EAMS-CBALM.....	52
3.6 Executing Sprint (ES).....	53
3.6.1 Acceptance Tests for the Selected BDDUSS of EAMS-CBALM.....	54
3.6.2 Implementation of the Selected BDDUSS of EAMS-CBALM.....	55
3.7 Sprint Review and Retrospective (SRR).....	56
3.7.1 Sprint Review and Retrospective for EAMS-CBALM.....	57
3.8 Final Considerations.....	58
CHAPTER 4 - ScrumOntoBDD Analysis	59
4.1 Action Research	59
4.2 Objectives, Settings, and Participants	61
4.3 Experimental Analysis	63
4.3.1 Action Planning	64
4.3.2 Action Taking	66
4.3.3 Evaluating.....	74
4.3.4 Specifying Learning	87
4.4 Final Considerations.....	91
CHAPTER 5 - RELATED WORKS	93
5.1 BDD Related Works.....	93

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

5.2 Ontology Related Works.....95

5.3 Final Considerations.....96

CHAPTER 6 - CONCLUSION.....98

6.1 Contributions and Limitations98

6.2 Future Works99

REFERENCES101

APPENDIX A THE WRITTEN INFORMED CONSENT FORM112

APPENDIX B THE INTERVIEWS ORIGINAL SCRIPT114

APPENDIX C THE AUDIO-TAPED INTERVIEWS TRANSCRIPTIONS136

Chapter 1

INTRODUCTION

This chapter presents an introduction to the theme of this MSc's dissertation, its motivations and objectives. Section 1.1 addresses the context of this research; Section 1.2 deals with motivations and objectives; and Section 1.3 describes how the structure of this document is organized.

1.1 Context

Agile software development refers to iterative software development approaches, methodologies and methods, where requirements and solutions evolve through collaboration between customers and developers. Among them, the following stands out: Scrum (Schwaber, 2017) that prescribes sprint reviews, which are meetings involving the developer team and the Product Owner (PO) to plan and evaluate sprints; and Behaviour-Driven Development (BDD) (North, 2006), which prescribes the definition of usage scenarios (behaviour specifications) upfront as a way to better understand what the software is supposed to do (its 'behaviour').

The importance of the Ontology discipline is recognized in several research areas of Computer Science, and ontologies are being used in the design of Information Systems (IS) related to a diversity of knowledge fields, such as

Languages, Engineering, Health, and Education. According to Guarino (Guarino, 1998), an ontology can impact an IS in a *temporal dimension*, concerning whether it is used at the IS development or run times, and in a *structural dimension*, concerning the way it affects the main IS components, i.e., application programs, databases, and user interfaces.

The Medicine Programme of the Federal University of São Carlos (UFSCar), located in São Carlos city at São Paulo (SP) state in Brazil, was established in 2006, and follows a socio-constructivist educational approach, has a competency-oriented pedagogical programme, and employs active learning methodologies, such as Problem Based Learning (PBL) (Rhem, 1998) and Practice Based Learning (Carlisle, 2009).

To provide computational support for courses that employ these methodologies, the Ubiquitous Computing Group (UCG) of UFSCar developed, in partnership with the UFSCar Medicine Department (DMed), the Teaching and Research Institute (TRI) of the Sírio-Libanês Hospital (SLH), and a software house, the Learning Management System (LMS) called Educational and Academic Management System for Courses Based on Active Learning Methodologies (EAMS-CBALM) (Santos, 2016).

1.2 Motivations and Objectives

EAMS-CBALM was developed using Scrum, and during its development, it was quite often necessary to redefine some system behaviour scenarios and, consequently, the Product Backlog (PB) items, due to misunderstanding of the stories reported by the PO. The definition of test suites was also cumbersome, resulting in test suites that were incomplete or did not at all comply with the system requirements.

In order to deal with the mentioned problems, this MSc's project answered two main research questions:

RQ₁ - How to improve the communication between PO and developers?

RQ₂ - How to eliminate the ambiguities intrinsic of using natural languages to report user stories?

Based on these questions, we established two main hypotheses:

H₁ - Combining BDD with Scrum can improve this communication.

H₂ - Employing ontologies can eliminate these ambiguities.

To verify the first hypothesis (H₁), we developed a case study in the context of the EAMS-CBALM, by employing a ubiquitous language for the Education domain together with BDD scenarios and acceptance tests, in order to allow the PO to follow and properly communicate with the developers throughout the development process. This work was reported in Souza (Souza, 2017).

To verify the second hypothesis (H₂), we developed another case study in the context of the EAMS-CBLAM, by employing domain ontologies to describe the UFSCar Medicine Programme, in order to reduce the ambiguities caused by using a natural language as a ubiquitous language. This work was reported in Souza (Souza, 2018).

The main objective of this MSc's dissertation is to join these two works by means of an approach, which combines Scrum, Ontology and BDD, for assuring both hypotheses. This approach was named ScrumOntoBDD.

1.3 Structure of the Dissertation

After this introduction, this MSc's dissertation is organized as follow:

Chapter 2 deals with the knowledge areas addressed in our research, i.e., Agile Software Development, Ontologies, Active Learning Methodologies, and Learning Management Systems, focusing on the relevant matters of these areas for our MSc's project.

Chapter 3 presents ScrumOntoBDD, our proposed approach for agile software development, which combines Scrum, Ontology and BDD. First, an overview of this approach is provided, and then its main activities are detailed and exemplified with case studies related to the EAMS-CBLAM development.

Chapter 4 describes an experimental analysis of ScrumOntoBDD, whose main objective is to evaluate our two hypotheses when employing this approach. After an introduction to Action Research, the process used for this analysis, the experiment is described according to the Action Research cycle and phases, and its results are discussed in regard of our hypotheses.

Chapter 5 discusses some related works to our work, splitting them into two groups according to the systematic literature reviews carried out during the development of this MSc's project: the first employing BDD for improving software development, and the second employing Ontology also for improving software development.

Finally, Chapter 6 gives our concluding remarks and the directions for future works, mainly based on the ScrumOntoBDD analysis performed on Chapter 5.

Chapter 2

THEORETICAL AND TECHNICAL BACKGROUND

For the development of this MSc's project, it was fundamental to acquire knowledge in the areas of Agile Software Development, Ontologies, Active Learning Methodologies, and Learning Management Systems. Section 2.1 addresses the first area, dealing mainly with Scrum and BDD; Section 2.2 addresses the second area; Section 2.3 addresses the third area, dealing mainly with UFSCar Medicine Programme; Section 2.4 addresses the fourth area, dealing mainly with EAMS-CBLAM; and Section 2.5 has the final considerations.

2.1 Agile Software Development

Nowadays, it is difficult to predict how a software-based system should evolve as market conditions and customer needs change rapidly, and new technologies are constantly emerging. In order to help software developers to deal with these problems, agile methods have been proposed. The main ideas behind these methods can be found in the *Manifesto for Agile Software Development* (Beck, 2001):

“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

That is, while there is value in the items on the right, we value the items on the left more.”

By using agile methods, the software is not developed as a single unit, but as a series of increments, each increment including some new system features. New releases of the system, corresponding to the produced increments, should be periodically available to the customers, and this period may vary from one week to one month depending on the system size and complexity.

Agile methods are usually employed to develop systems where their requirements can quickly change during the development process. Therefore, it is crucial to involve customers in this process so they can propose changes and new requirements for the next system increments.

The well-known agile methods include Extreme Programming (Beck, 2012), Crystal (Cockburn, 2004), Adaptive Software Development (Highsmith, 2000), DSDM (Stapleton, 2003), Feature Driven Development (Palmer, 2002), and Scrum (Schwaber, 2017).

2.1.1 Scrum

Scrum is an agile software development method that emphasizes the interactions between users and developers, mainly when the requirements are established and selected as *user stories*, with the objective to obtain a fast delivery and a satisfactory quality of the product. Most of the clarifications and details regarding the product under development are obtained during these interactions, allowing product adaptations to be done quickly, thus avoiding bottlenecks and delays and reducing the likelihood of unsatisfactory results.

In Scrum, the units of work are divided in *sprints*, which can last between a week and a month, period in which the developers creates an increment of the product to be delivered to the customer. According to Rubin (Rubin, 2012) and as illustrated in Figure 1, Scrum is a framework for organizing and managing team work, and Scrum practices involve specific roles, events, and artefacts.

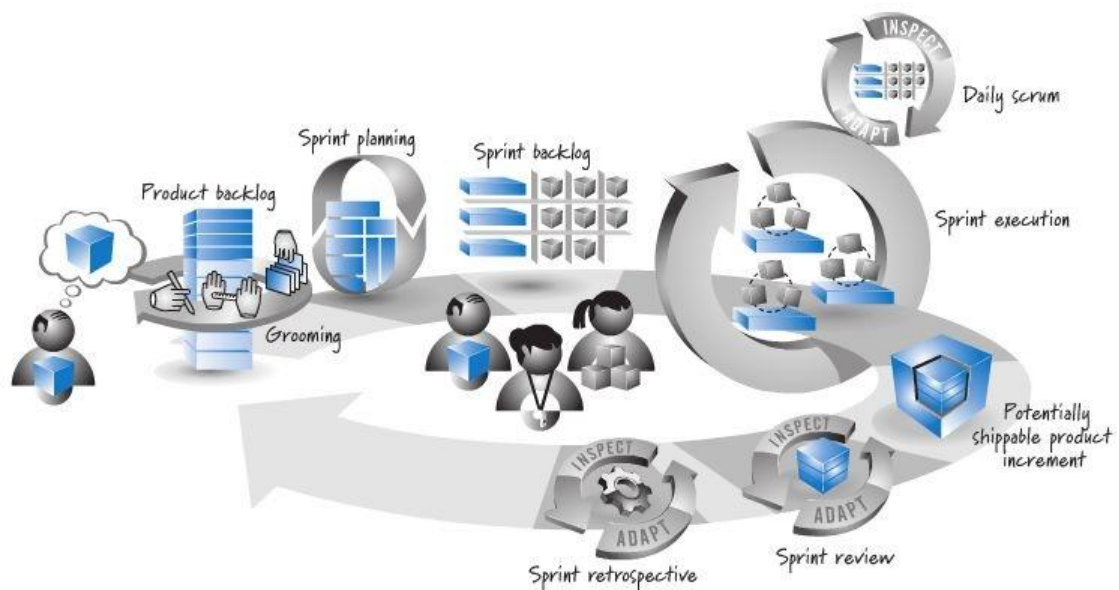


Figure 1: Scrum roles, events, and artefacts adapted from Rubin (Rubin, 2012)

The development of a product by employing Scrum may involve one or more Development Teams (DT), each one composed at least of three Scrum roles:

- (a) *Product Owner (PO)*, who can be designated by the customer, is responsible for what will be developed and in what order, and for tracking product development;
- (b) *Scrum Master (SM)* is responsible for promoting and supporting Scrum by helping the team understand its theory, practices, rules, and values; and
- (c) *Scrum Team (ST)* is composed by professionals responsible for delivering at each sprint a usable increment of the product requested by the customer.

According to Schwaber (Schwaber, 2017), “Scrum’s roles, events, artefacts, and rules are immutable and although implementing only parts of Scrum is possible, the result is not Scrum. Scrum exists only in its entirety and functions well as a

container for other techniques, methodologies, and practices.” During the Scrum development cycle the following main artefacts are produced:

(a) *Product Backlog (PB)* is a prioritized list of requirements stated as user stories, broken down (*grooming*) into a set of features, that reflects the PO vision of the product to be created;

(b) *Sprint Backlog (SB)* is a list containing a subset of PB items, followed by the tasks to be performed for each item, which the ST believes and commits to complete on that sprint; and

(c) *Product Increment (PI)* is a potentially shippable increment of the product, representing part of the PO's product vision, which is produced by ST at the end of sprint execution, and can be released to the customer.

To produce these artefacts, the following main events take place:

(a) *Sprint Planning* is a meeting involving SM and ST for planning the work to be performed in that sprint, which can last a maximum of eight hours for a one-month sprint or less for shorter sprints;

(b) *Sprint Execution*, where the ST guided by the SM performs the needed tasks to get the selected features done, and to produce a potentially shippable PI;

(c) *Daily Scrum* is a ST 15-minute meeting held every day of the sprint for planning the work within the next 24 hours, which allows for optimizing team collaboration and performance as the work produced since the last daily scrum is scrutinized, and sprint's next work is predicted;

(d) *Sprint Review* is a DT meeting held at the end of the sprint, which can last a maximum of four hours for a one-month sprint or less for shorter sprints, for inspecting the PI produced on that sprint, and for adapting the PB if needed; and

(e) *Sprint Retrospective* is a meeting held after the Sprint Review and prior to the next Sprint Planning, which can last a maximum of three hours for a one-month sprint or less for shorter sprints, where the DT inspect itself and create an improvement plan to be applied during the next sprint.

Recently, Dingsoyr (Dingsoyr, 2016) conducted a survey to visualize the trends of some development practices in the last ten years. To do that, they investigated the interest from developers and researches about the following topics from 2006 until 2016: Scrum, Extreme Programming, DevOp¹ and Continuous Integration².

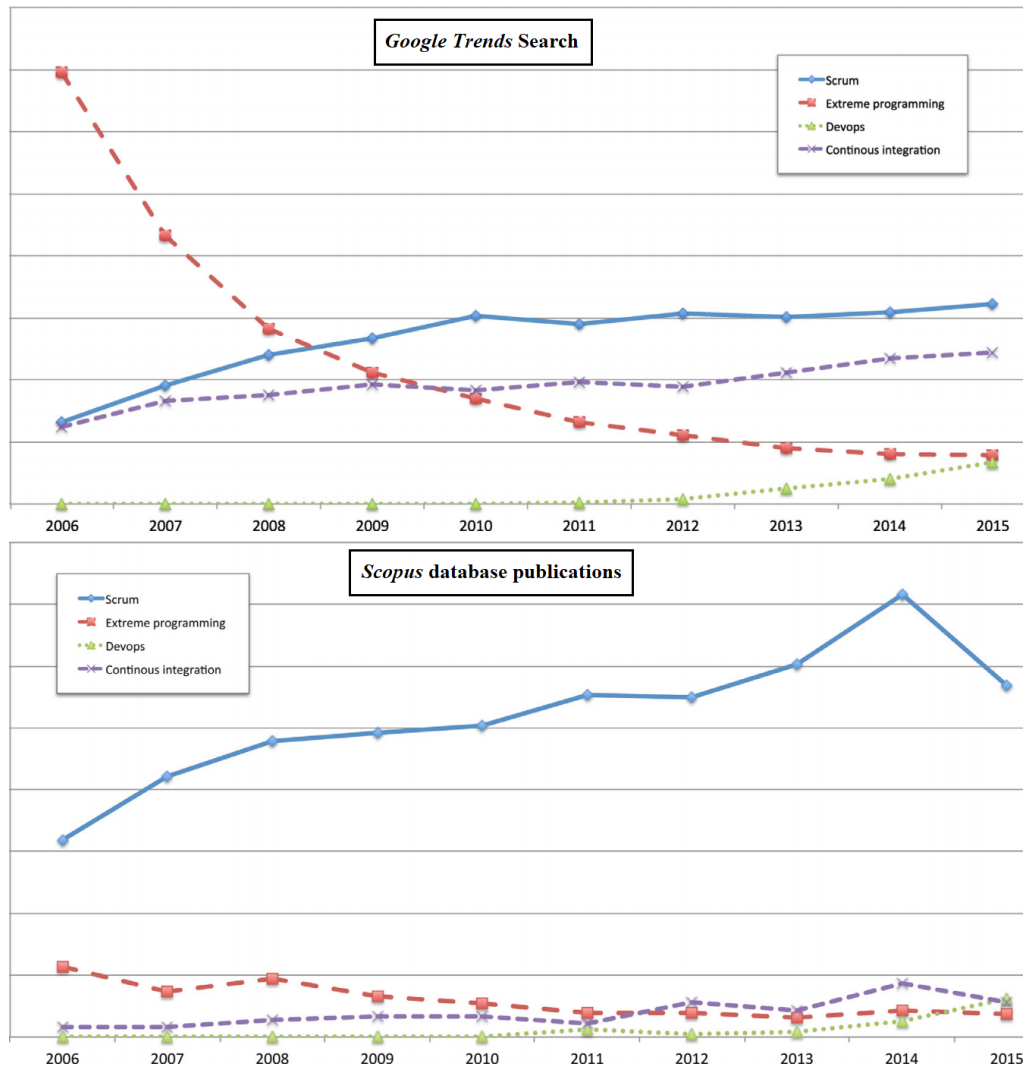


Figure 2: A the top chart, Google Trends research. At the botton chart, Scopus publications database (from(Dingsoyr, 2016))

¹ DevOps can be understood as an interdisciplinary community of practices dedicated to the study of the construction and evolution of resilient systems of rapid change in scale. Source: <<http://theagileadmin.com/what-is-devops/>>

² Continuous integration is a software development practice in which software is continuously integrated during development. Continuous integration requires at least daily integration and automatized compile and tests verification (Dingsoyr, 2016)

In the top chart of Figure 2, the interest in the past 10 years between developers based on Google Trends, and in the bottom chart, the interest at the same period among researchers based on the Scopus publications database. Scrum stands out and its interest and use has been increasing over time. Probably, the decline in 2015 among researchers is related to the late indexing of articles in the database (Dingsoyr, 2016).

2.1.2 Behaviour-Driven Development (BDD)

Test Driven Development (TDD) is a software evolutionary development methodology, based on short development cycles, in which automated tests are described previously to the production of code (Beck, 2002). Acceptance Test Driven Development (ATDD) is a TDD variation, where software development is driven by acceptance tests, which are used to represent stakeholder's requirements (Koskela, 2008).

Behaviour-Driven Development (BDD) is an agile software development methodology, originally proposed by North (North, 2006) and considered as an evolution of TDD and ATDD, whose fundamental principle is: "stakeholders and developers should refer to the same system in the same way". For achieving this goal, a ubiquitous language is required that is understandable by all those involved in system development and enables executable granular specifications of the system's behaviour and testing (Diepenbeck, 2014).

Six main characteristics of BDD were identified in Solis (Solis, 2011): ubiquitous language; iterative decomposition process; user story and scenario templates; automated acceptance testing with mapping rules; readable behaviour-oriented specification code; and behaviour-driven at different phases. Using these characteristics, Solis also analyses seven of the main BDD tools: NBehave (NBehave, 2011) and JBehave (JBehave, 2015); MSpec (MSpec, 2008) and RSpec (RSpec, 2016); StoryQ (StoryQ, 2010); Cucumber (Cucumber, 2014); and SpecFlow (SpecFlow, 2016). We reanalyzed all seven BDD toolkits to verify the BDD characteristics. We reanalyzed all seven BDD toolkits to verify the BDD

characteristics. This new analysis is summarized in Tables 1 and 2 that shown BDD characteristics and languages supported by these tools.

Supported BDD characteristics		xBehave Family		xSpec Family		StoryQ	Cucumber	SpecFlow
Ubiquitous language definition		X	X	X	X	X	X	X
Iterative decomposition process		X	X	X	X	X	X	X
Editing plain text based on	User story template	√	√	X	X	X	√	√
	Scenario template	√	√	X	X	X	√	√
Automated acceptance testing with mapping rules		√	√	X	X	X	√	√
Readable behaviour oriented specification code		√	√	√	√	√	X	√
Behaviour driven at different phases	Planning	X	X	X	X	X	X	X
	Analysis	√	√	X	X	X	√	√
	Implementation	√	√	√	√	√	X	X

Table 1: Supported BDD characteristics by the analyzed BDD tools

	xBehave Family		xSpec Family		StoryQ	Cucumber	SpecFlow
	JBehave	NBehave	RSpec	MSpec			
Supported programming languages	Java	C#	Ruby	C#	C#	Ruby, Java, Groovy, C#, etc	C#

Table 2: Supported programming languages by the analyzed BDD tools

BDD is an evolving methodology that does not have a clear definition and unanimous understanding, and the existing support tools focus mainly on the implementation phase, providing limited support to the requirements gathering, analysis, and design phases of software life cycle. Starting from Solis (Solis 2011), we did a systematic review of BDD-related work, and an analysis of the current BDD toolkits. Figure 3 shows the BDD process depicted using the Structured Analysis and Design Technique (SADT) diagrammatic notation (Ross, 1977).

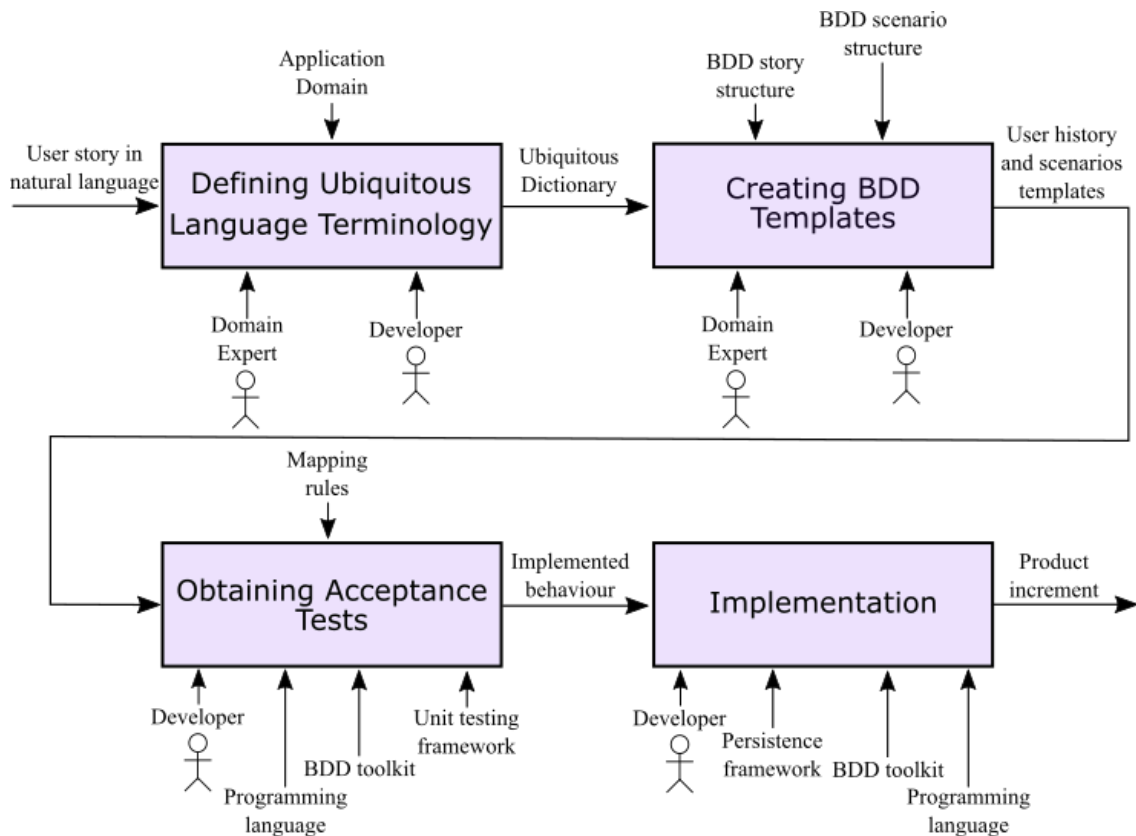


Figure 3: The BDD process in SADT based on the systematic review and analysis of BDD toolkits

2.1.2.1 Ubiquitous Language

A ubiquitous language is essential in BDD. It must have a structure derived from a business domain model, offer a terminology that is understandable to both customers and developers, and be used in all system development phases (Evans, 2003). Therefore, a ubiquitous language must be created for the communication between the stakeholders and developers.

Most of the ubiquitous language terminology should be defined in the analysis phase, but new terms can be added at any time and at any phase. In the design and implementation phases, this terminology is used to name classes and methods, making the code clearer and better readable. BDD itself has a simple pre-defined ubiquitous language for the analysis phase, which is domain-independent. This language is used to formulate the user stories and scenarios.

2.1.2.2 BDD Templates

The BDD analysis phase starts by identifying the most expected system behaviours, which are derived from the business outcomes to be produced by the system. Based on these business outcomes, feature sets are defined, where each feature set can contain subsets, and indicates what should be accomplished to achieve a specific business outcome.

This initial analysis may be sufficient for a first implementation, even if important system behaviours are yet undisclosed. However, as new behaviours are revealed, the whole process illustrated in Figure 2 can be performed again, thus characterizing the "iterative decomposition process" described in Solis (Solis, 2011).

Features and their acceptance criteria are expressed through user stories, and describe the interactions between a user and the system. A user story should elucidate the user's role in this story, the feature desired by the user, and the benefit gained by the user if the system provides the desired feature. Due to different contexts, a user story may have different versions that will lead to different story instances, called scenarios, which in turn should describe specific contexts and outcomes of this user story.

A scenario describes how the system that implements a given feature should behave when it is in a main context with possible additional contexts that could be represented by additional conditions. When an event happens, e.g., a system entry, the scenario's result can be one or more actions that change the state of the system or produce some system output.

BDD user stories and scenarios follow the predefined templates described in North (North, 2007) by employing a simple ubiquitous language. According to these templates, a User Story is described with a **Title**, a **Narrative** and a set of **Scenarios** representing the **Acceptance Criteria**. The Title provides a general description of the story, making reference to a feature that this story represents. The Narrative is divided in three clauses: (i) **In order to** describe an activity to be performed by a user; (ii) **As a** correspond to the role played by the user on that story; and (iii) **I want** show the benefit obtained by the user once the activity is performed. The Acceptance Criteria are defined through a set of Scenarios, each one with a Title and three main

clauses: (i) **Given** to provide main context that could be represented by a system state; (ii) **When** to describe a specific event that will trigger the Scenario; and (iii) **Then** to present the main outcome that could be an action for changing the system. Each one of these clauses can include an **And** statement to provide multiple contexts, events and/or outcomes. Each statement in this representation is called step. These templates are illustrated in Figure 4.

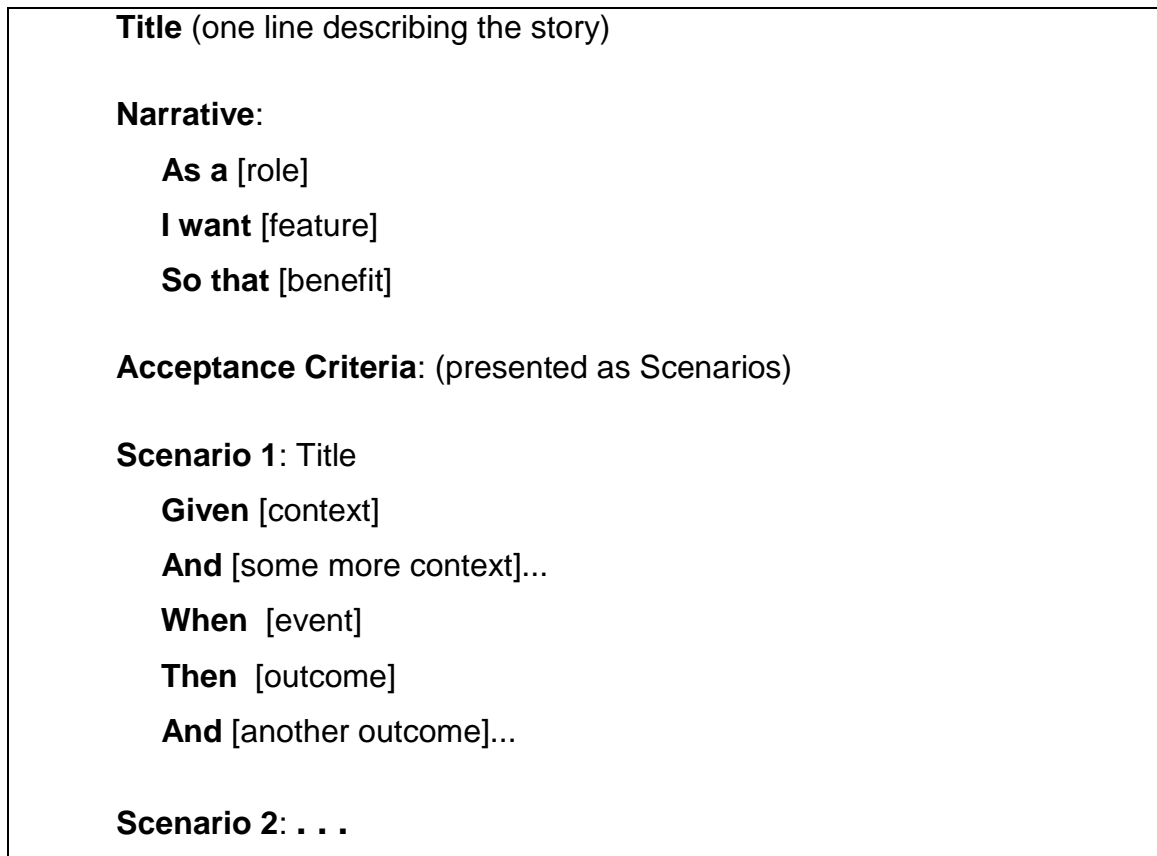


Figure 4: BDD templates for user story and scenarios (from (North, 2017))

2.1.2.3 Acceptance Tests

A BDD acceptance test is an executable specification of the system behaviour, which verifies the interactions or behaviours of objects rather than their states. The produced scenarios are translated into tests that guide the implementation. A scenario is composed of several steps, where each step is an abstraction that represents the three elements of a scenario: context, event and action. The meaning of these elements is: in a particular case of a user history or context *C*, when the event *X* happens, the system response should be *Z*. Each step is mapped to one test

method, and the scenario passes only if all its steps pass. Each step follows the TDD's process: red, green, and refactoring to make it pass.

Since in BDD all scenarios must be executed automatically, the acceptance criteria must also be imported and analysed automatically. The classes that implement the scenarios read their specifications, which are written in the ubiquitous language, and execute them. Therefore, the mapping between scenarios and testing code needs to be explicitly defined.

2.1.2.4 Implementation

In BDD the code must be readable, contain the specification, describe the behaviour of the objects and be part of the system documentation. The classes and methods names must be sentences, and the names of the methods must indicate their functionality. Mapping rules assist in the production of readable behaviour-oriented code and ensures that classes and methods names are the same as user story titles and scenarios, respectively. In addition, these names make use of the ubiquitous domain-specific language defined in the project.

2.1.3 Combining Scrum and BDD

According to Chauhan (Chauhan, 2016) and as illustrated in Figure 5, BDD can be employed in the following Scrum artefacts and rituals:

(a) *Product Backlog* and *Sprint Backlog*, BDD provides a common ground and language for transforming requirement into functional behaviour, and for changing functional behaviour into technical specification to be implemented;

(b) *Daily Scrum* meetings, BDD provides a common platform for understanding the development of each activity, and for avoiding any confusion and knowledge gap among the developers for producing the final implementation; and

(c) *Sprint Planning* meetings, BDD provides valid documents for discussing the final product behaviour in user acceptance, since BDD is in place from sprint planning and all stakeholders were involved when defining requirement.

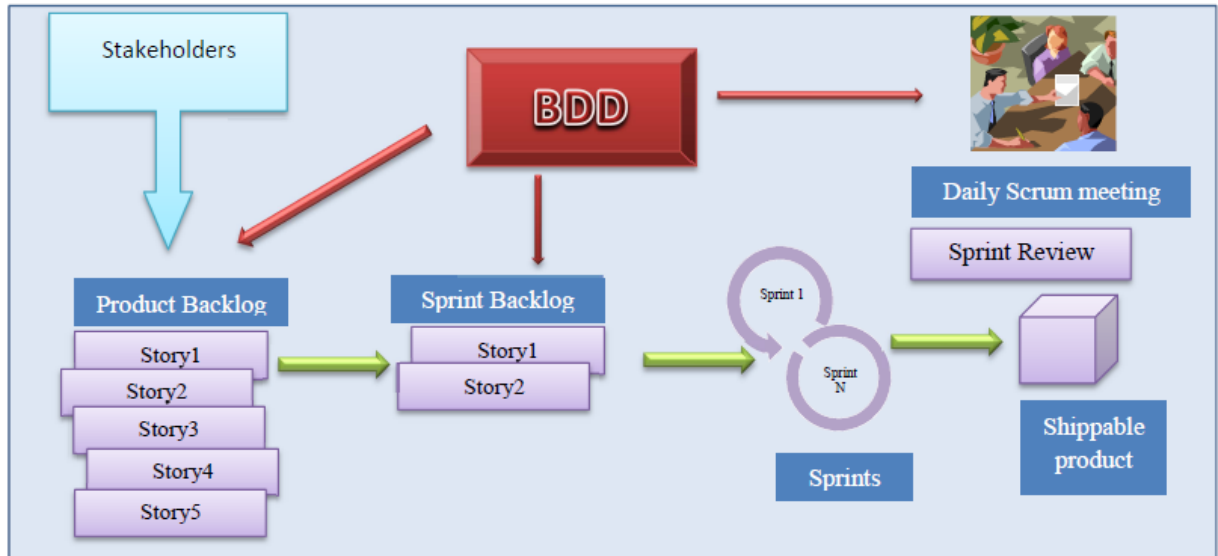


Figure 5: Points for connecting BDD to Scrum (from (Chauhan, 2016))

Therefore, the combination of Scrum with BDD allows for a better visibility of what was required, why was needed and who has requested. As stated in Chauhan (Chauhan, 2016), this combination brings the following benefits: avoids misinterpretation of user stories; transform requirement into functional behaviour of the software; provide a common platform and vocabulary to understand requirement in better way; development team is aware of User Acceptance behaviour in advance for all user stories; as requirements (user stories) are inter-dependent, therefore helps in avoiding any confusion among development team; and spring meetings and daily scrum meetings are more effective.

2.2 Ontologies

The term ontology was introduced by philosophers in the seventeenth century as the philosophical study of nature and relationships of being. He appeared at a computer-related conference for the first time in 1967 (Mealy, 1967), and has since been employed in the fields Computer Science and Information Science.

On these fields, according to Gruber (Gruber, 1995), an ontology is “an explicit specification of a conceptualization”, where conceptualization is composed by “the

objects, concepts, and other entities that are presumed to exist in some area of interest and the relationships that hold among them”.

Futhermore, according to Gruber (Gruber, 2008), “an ontology specifies the concepts, relationships, and other distinctions that are relevant for modelling a domain, and such specification takes the form of the definitions of representational vocabulary (classes, relations, and so forth), which provide meanings for the vocabulary and formal constraints on its coherent use”.

In Computer Science, ontologies have been used mainly in the following areas (Smith, 2001): Database and Information Systems; Artificial Intelligence; and Software Engineering. In the Artificial Intelligence context, ontology defines a set of representational terms, associating the names of entities in the universe of discourse (e.g., classes, relations, functions) with natural language texts that describe the meanings of these names, and with formal axioms that constrain the interpretation and the use of these terms.

2.2.1 Ontology Classification

There are several ontology classifications in the literature. As stated in Guarino (Guarino, 1998) and illustrated in Figure 6, ontology can be classified according to its generality and dependence levels into:

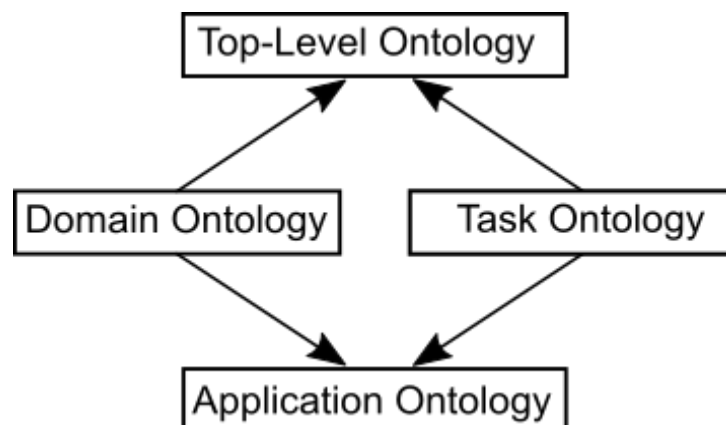


Figure 6: Ontology classification (from (Guarino, 1998))

(a) Top-level ontology, also called upper or foundation ontology, describes very general concepts (e.g., Object, Property) that are independent of a particular domain;

(b) Domain ontology describes the vocabulary related to a generic domain (e.g., Medicine, Course) by specializing terms introduced in the top-level ontology;

(c) Task ontology describes the vocabulary related to a generic task (e.g., Diagnosing, Lecturing), by also specializing terms introduced in the top-level ontology; and

(d) Application ontology describes concepts depending on a particular domain and task (e.g., the *roles* played by domain entities while performing the activity of *given a lecture*), which are specializations of the related ontologies.

A top-level ontology aims at supporting semantic interoperability among domain ontologies, by providing a common basis for formulating definitions. The reuse and maintenance of systems supported by domain ontologies usually requires a merge of these ontologies, and if they are derived from the same high-level ontology, this merging can be performed automatically.

Unfortunately, most of the available domain ontologies are not derived from the same top-level ontology. Moreover, different domain perceptions, different usage intentions, and different languages, give rise to specific application ontologies in the same domain that are often incompatible. In order to deal with these problems, application ontologies can be constructed from a domain reference ontology.

In this regard, the Higher Education Reference Ontology (HERO) is proposed in Zemmouchi-Ghomari (Zemmouchi-Ghomari, 2013) for providing a consensual knowledge of the university domain, and to be shared and reused among several stakeholders. HERO describes several aspects of the university domain, such as organizational structure, staff, and income, which can be used to derive more elaborate domain ontologies for higher education. Figure 7 illustrates the key concepts of the HERO ontology.

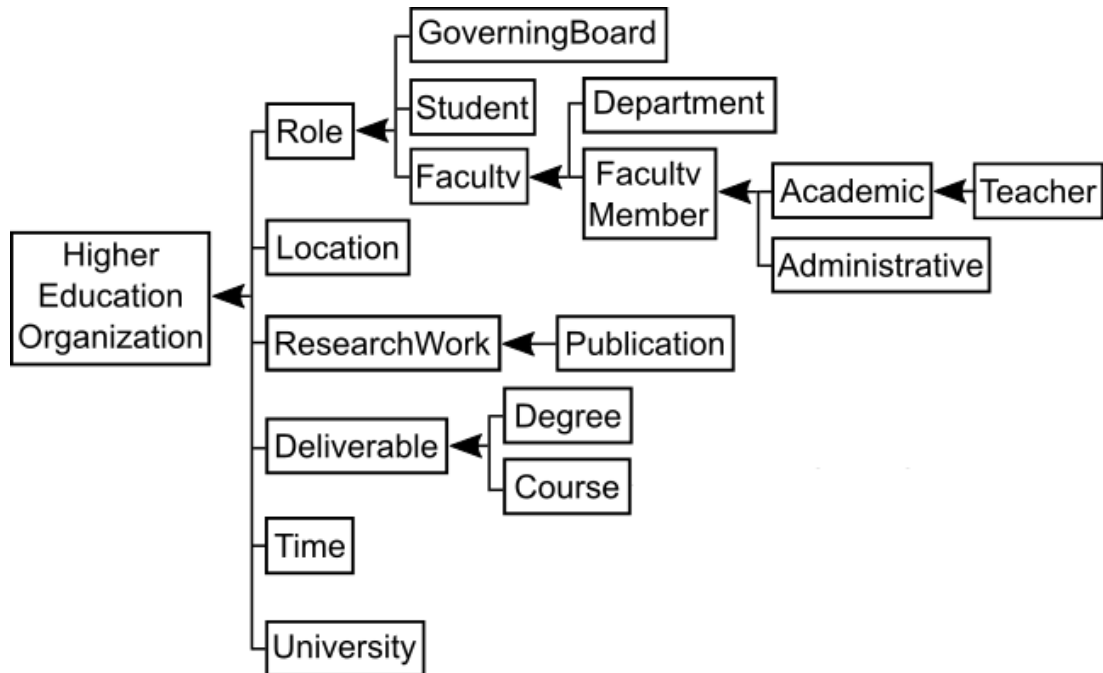


Figure 7: Key concepts of HERO ontology adapted from (Zemmouchi-Ghomari, 2013))

2.2.2 Ontology Languages

In the last two decades, several languages have been developed for ontology formal description, allowing the knowledge representation related to specific domains, and often including reasoning rules for processing this knowledge. According to Berners-Lee (Berners-Lee, 2009), “a knowledge representation system must have the following properties: a reasonably compact syntax; a well defined semantics so that one can say precisely what is being represented; sufficient expressive power to represent human knowledge; an efficient, powerful, and understandable reasoning mechanism; and must be usable to build large knowledge bases.”

Since a major focus has been given to ontology languages for Web, and since the Web is decentralized, such languages should enable the definition of several vocabularies and their evolution. As stated in Gutierrez-Pulido (Gutierrez-Pulido, 2006), “a Semantic Web language must describe meaning in a machine-readable way. Therefore, an ontology language needs not only to include the ability to specify vocabulary, but also the means to formally define it in such a way that it will work for automated reasoning.”

Ontology languages can be classified into two major categories, *traditional* and *Web-based* (Su, 2002). In the first category, there are languages based on: First-Order Predicate Logic, such as *Knowledge Interchange Format (KIF)* (Genesereth, 2004) and *Cycl* (Cycorp, 2014); Frame, such as *Frame Logic (F-logic)* (Kifer, 1995) and *Operational Conceptual Modelling Language (OCML)* (Motta, 1998); and Description Logic (DL), such as *Loom* (MacGregor, 1991). In the second category there are Web standards and Web-based ontology languages compatible with Web standards.

Examples of Web standards are: *EXtensible Markup Language (XML)* (Bray, 2008) was developed by the World Wide Web Consortium (W3C) for encoding documents and data on the Web that are readable by human and machine; and *Resource Description Framework (RDF)* (Gandon, 2014) was also developed by W3C for describing Web resources and allowing the semantics specification of data based on XML in a homogeneous and interoperable way.

Examples of Web-based ontology languages are: *Simple HTML Ontology Extensions (SHOE)* (Luke, 2000) is a HTML extension for incorporating semantic knowledge in Web pages by annotating them, and for providing categories, relations, inference rules, and rules to specify ontologies; *DARPA Agent Markup Language + Ontology Interchange Language (DAML+OIL)* (Connolly, 2001) is a semantic markup language proposed as a W3C standard for ontological and metadata representation, which includes aspects from OIL (Fensel, 2000) and is built on RDF Schema (RDFS) (Brickley, 2014); and *Web Ontology Language (OWL)* (W3C, 2012) is the W3C standard ontology language for the Semantic Web.

OWL was created in 2001 by a W3C Working Group (WG) as an RDS extension compatible with previous ontology languages (e.g., SHOE, DAML+OIL), but more powerful from them for expressing semantics. It includes conjunction, disjunction, existentially, and universally quantified variables for doing inferences and deriving knowledge.

There are two versions of OWL: OWL1 that was developed by the W3C Web Ontology WG (W3C Web Ontology WG, 2004), and OW2 that was developed by the W3C OWL WG (W3C OWL WG, 2012). OW1 includes the classes OWL Full, OWL

DL and OWL Lite. OWL2 is the latest OWL version, keeps OWL Full and OWL DL, adds richer datatypes, qualified cardinality restrictions, reflexive, asymmetric and disjoint properties, and defines different profiles: OWL 2DL, OWL 2EL, OWL2 QL, and OWL 2RL.

We employed OWL in this MSc's project for specifying our ontologies since this language is Web-based, is a standard for the Semantic Web, is compatible with several other ontology languages, and is used by a large community in many contexts. Figure 8 depicts an excerpt of HERO ontology in OWL showing some key concepts of this ontology.

```

<!-- http://www.UniversityReferenceOntology.org/HERO#Role -->
  <owl:Class rdf:about="http://www.UniversityReferenceOntology.org/HERO#Role">
    <rdfs:subClassOf rdf:resource="&owl;HigherEducationOrganization"/>
  </owl:Class>
<!-- http://www.UniversityReferenceOntology.org/HERO#Location -->
  <owl:Class rdf:about="http://www.UniversityReferenceOntology.org/HERO#Location">
    <rdfs:subClassOf rdf:resource="&owl;HigherEducationOrganization"/>
  </owl:Class>
<!-- http://www.UniversityReferenceOntology.org/HERO#ResearchWork -->
  <owl:Class rdf:about="http://www.UniversityReferenceOntology.org/HERO#ResearchWork">
    <rdfs:subClassOf rdf:resource="&owl;HigherEducationOrganization"/>
    <rdfs:isDefinedBy xml:lang="en">Faculty research work comprises a variety of related
      activities, such as plan and perform studies and experiments, [...]</rdfs:isDefinedBy>
  </owl:Class>

```

Figure 8: Excerpt of HERO ontology in OWL

2.2.3 Ontology Tools

When starting out an ontology development, either from scratch or by reusing existing ontologies, it is very important to look for appropriated tools. Although there are many free and commercial ontology tools that offer different functionalities, most of them do not allow the ontology developers to interchange their ontologies from one tool to another.

Ontology tools can be employed in the creation, implementation, and maintenance phases of the ontology life cycle, helping to acquire, organize, and visualize domain knowledge before and during the ontology development. In

particular, ontology editors allow users to manipulate, inspect, browse, and code ontologies, and special attention has been given in recent years to the Semantic Web editors responsible for creating and manipulating ontologies (Alatrish, 2013).

Several ontology editors related to the Semantic Web can be found on the Internet, and some of them are used by a large number of people. Among the most requested of these editors stand out: Apollo (KMI, 2004), Semantic Web Ontology Overview and Perusal (SWOOP) (MIND SWAP, 2004), TopBraid Composer (TopQuadrant, 2011), and Protégé (Musen, 2015). Apollo, SWOOP and Protégé are open source, and TopBraid Composer (FE) is under software license.

Apollo was developed by the Knowledge Media Institute (KMI) of The Open University, and allows users to model ontology with basic primitives, such as classes, instances, functions, and relations. Its internal model is a frame based on the Open Knowledge Base Connectivity (OKBC) protocol, and its knowledge base is a hierarchical organization of ontologies. Each ontology can inherit other ontologies and use their classes, and every ontology inherits at least a default ontology that contains all primitive classes (e.g., Boolean, integer, float, string, list). Each class can create several instances that inherit all slots from that class, and each slot consists of a set of facets. According to KMI (KMI, 2004), Apollo has the following main features: full consistency check during editing; open design based on views; special dialog for quick creation of anonymous instances; I/O plug-in architecture - export plug-ins to CLOS and OCML; and Java based user-interface.

SWOOP was developed by the Maryland Information and Network Laboratory (MIND) in the Semantic Web and Agents Project (SWAP) at the University of Maryland, and allows users to create, edit, and debug OWL ontologies, offering them an interface with hyperlinked capabilities and several OWL presentation views. It has reasoning support, and provides a multiple ontology environment, by which entities and relationships across various ontologies can be compared, edited and merged. According to MIND SWAP (MIND SWAP, 2004), SWOOP has the following main features: it is simple to load ontologies from the Web and to navigate within and between them; multiple ontologies may be loaded at the same time; ontologies, classes, properties, and individuals are rendered in a high level, accessible manner;

one can "view the source" of ontologies and their entities in a number of common syntaxes (e.g., RDF/XML, OWL Abstract Syntax, Turtle); OWL reasoners can be integrated for subsumption, consistency checking, etc., and default reasoners include a RDFS-like simple reasoner and Pellet, a Description Logic Tableaux Reasoner; ontology change management with extensive rollback and undo mechanisms; Share Annotations on Ontologies using the Annotea Protocol, and also attach and distribute Ontology Change sets with Annotations; search across multiple ontologies and 'find all references' of an OWL named entity; compare entities using a Resource Holder; Export Ontologies directly to a remote WebDav store.

TopBraid Composer was developed by the TopQuadrant Incorporation, and allows users to create and manage domain models and ontologies in RDF, RDFS, and OWL, offering them visual editing, and interoperability with Unified Modeling Language (UML) (OMG, 2017), XML Schema and databases. It is based on the Eclipse platform and Jena API, and integrates rule-based reasoning engines, offering a form-based user interface with the ability to view and edit ontologies in several serialization formats. Testing, consistency checking and debugging is supported by built-in OWL Inference engine, SPARQL query engine and Rules engine. It makes easier for an enterprise to move to Semantic Web standards by importing legacy models including XML Schemas, UML, RDB Schemas and spreadsheets. Open APIs are available, it can run with the database back-end for improve scalability. According to TopQuadrant (TopQuadrant, 2011), TopBraid Composer comes in three editions: Free Edition (FE) is an introductory version with only a core set of features; Standard Edition (SEd) includes all features of FE plus other features such as graphical viewers, import facilities, and advanced refactoring support; Maestro Edition (ME) includes all features of SE plus other features such as support for TopBraid Live, EVN and Ensemble as well as SPARQLMotion.

Protégé was developed by the Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine, and allows users to create, visualize, manipulate, and save ontologies in several formats, such as RDF, RDFS, and OWL. It can be customized to provide domain-friendly support for creating knowledge models and entering data, and it can be extended by a plug-in architecture and Java-based API for building knowledge-base tools and applications. Protégé allows the definition of classes, class hierarchy's variables, variable-value

restrictions, relationships between classes, and properties of these relationships. It comes with visualization packages, such as OntoViz and Jambalaya, for visualizing ontologies with the help of diagrams, and supports collaborative ontology editing as well as annotation of ontology components and ontology changes. It includes an interface for the Semantic Web Rule Language (SWRL) for mathematical and temporal reasoning, and adds Prolog-type reasoning rules. According to Musen (Musen, 2015), Protégé currently exists in a variety of frameworks, and is offered in Web-based and Desktop-based systems.

We employed Protégé in this MSc's project for supporting the development of our ontologies because it is a widespread platform recognized for its scalability and extensibility, which provides a tool set for building domain models and knowledge-based applications with ontologies. Furthermore, it is a free and open-source platform that has been continuously improved by Stanford University. According to Harith (Harith, 2005), Protégé is a killer application since it is a highly transformative technology that creates new markets and behaviour patterns.

Figure 9 illustrates a Protégé 5.2.0 screenshot of the HERO ontology, showing on the left-hand side the asserted class hierarchy, and on the right-hand a graphical tool for visualising and exploring the class hierarchy. called OWLViz. The asserted class hierarchy shows the subclass hierarchy that can be obtained from *subClassOf* assertions in the ontology. Children nodes in the tree are subsumed by their parent nodes. Any classes that do not have an asserted superclass will show up directly under *owl:Thing* (the root).

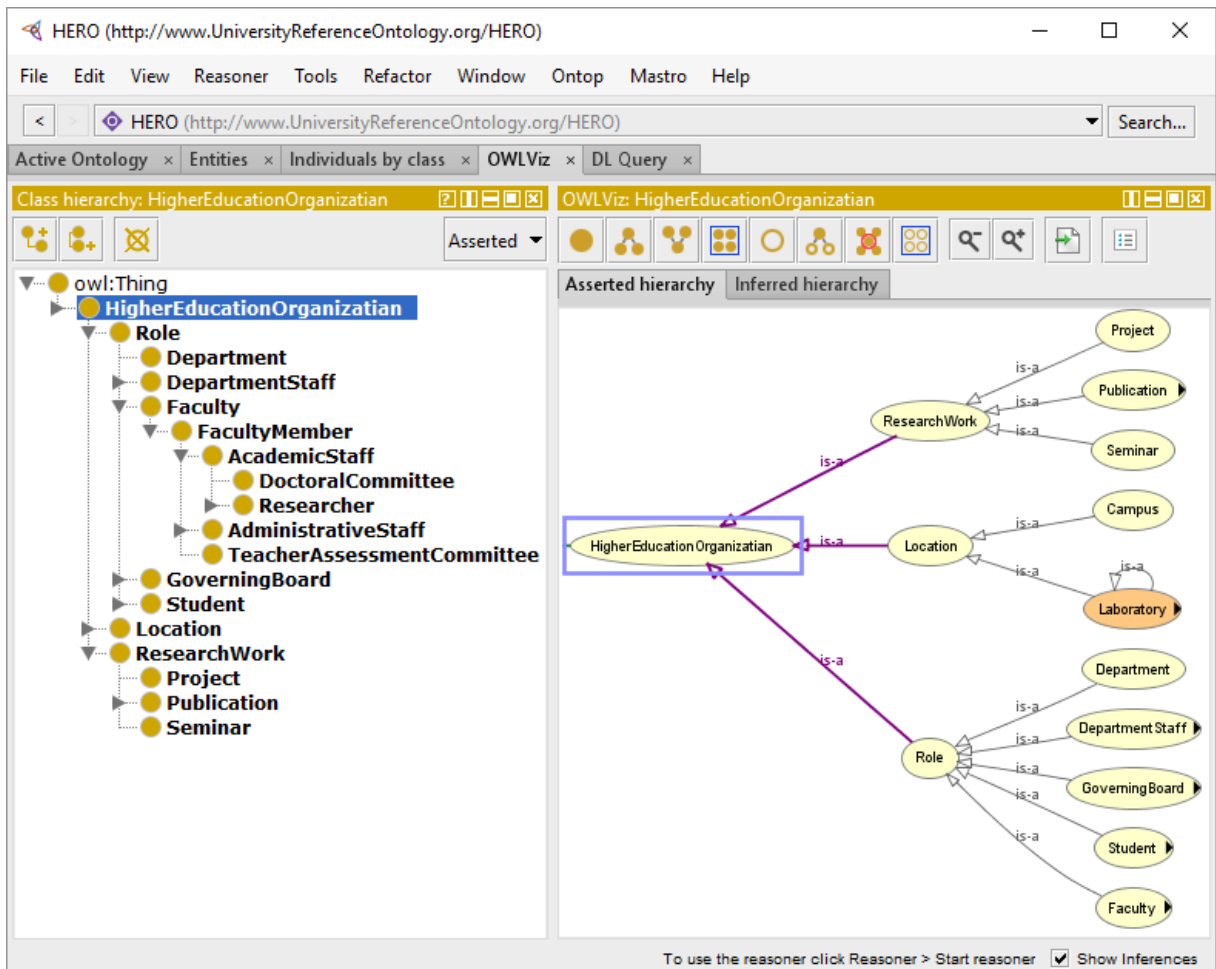


Figure 9: Protégé 5.2.0 screenshot of the HERO ontology

2.3 Active Learning Methodologies

The human being education extends throughout his life, and depending on the environment in which it happens it can be classified as: formal, when happens inside the classroom; non-formal, when happens outside the formal education system, including mentoring, continuing education, and professional training; and informal, when happens outside of any formal environment, such as in family, inside a community, and in daily work. Consequently, different teaching-learning processes may be involved in education, and they can occur anytime and anywhere (Chen, 2008).

With respect to the formal education in Brazilian universities, most of them employ traditional teaching-learning methodologies based on lectures classes. UFSCar is not an exception, but some of its programmes employ active learning methodologies. In particular, the UFSCar Medicine Programme, which was established in 2006, follows a socio-constructivist educational approach, has a competency-oriented pedagogical programme, and employs active learning methodologies, such as Problem Based Learning (PBL) (Rhem, 1998), and Practice Based Learning (Carlisle, 2009).

2.3.1 UFSCar Medicine Programme

Differently from traditional lecture courses, the UFSCar Medicine Programme curriculum is based on educational activities organized in three educational units as illustrated in Figure 10: Education Unit of Simulation of Professional Practice (EUSPP), Education Unit of Professional Practice (EUPP), and Education Unit of Elective Activities (EUEA). At the beginning of the programme the students have more EUSPP activities than EUPP and this is reversed throughout the programme (UFSCar, 2007).

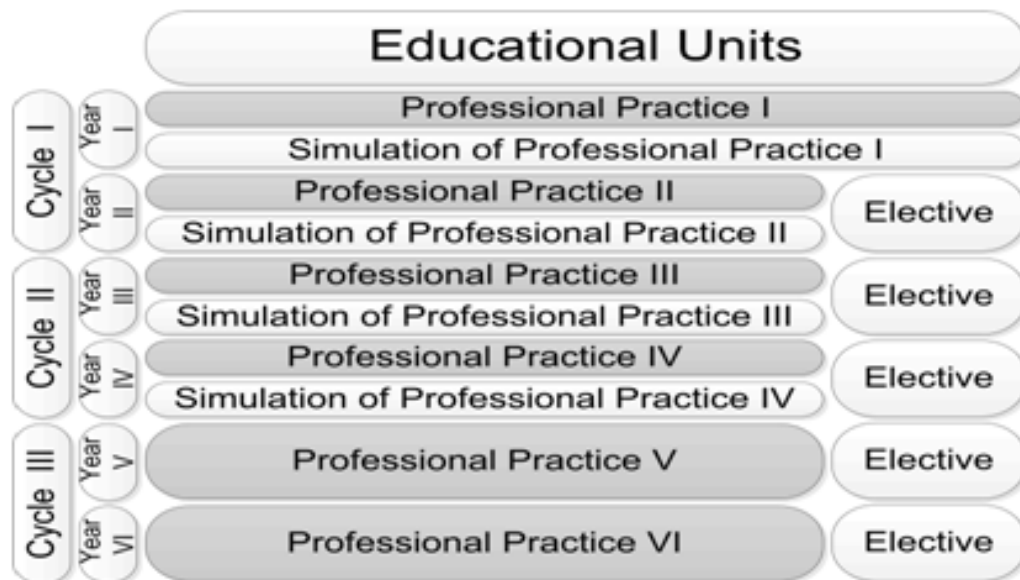


Figure 10: UFSCar Medicine Programme curriculum

The educational activities of the UFSCar Medicine Programme have learning triggers, which are problems simulating or portraying the daily activities to be

exercised by the graduates of this programme. These triggers activate the movements of the constructivist spiral illustrated in Figure 11, encouraging students to reflect, and stimulating them to develop their capacities. From each learning trigger, the students traverse all these movements, starting by identifying the problem, then formulating explanations, preparing learning questions, looking for new information, building new meanings, and finally evaluating the process.



Figure 11: Movements of the constructivist spiral

The Reflective Portfolio (RP) is a tool for recording and monitoring student activities, widely used in courses that employ active learning methodologies, since it is a flexible and evidence-based instrument that motivates the students to a continuous reflection, and a collaborative analysis of learning (Zubizarreta, 2009). In the UFSCar Medicine Programme, this portfolio is used in all student activities, both in the internal and external to the university campus, and at the beginning of this programme the media of this tool was 100% paper.

2.4 Learning Management Systems

The growing use of Information and Communication Technologies (ICT) in Education, combined with the dissemination of network technologies and the Internet, has made E-learning practices significantly evolve. According to Guri-Rosenblit (Guri-Rosenblit, 2005), “E-learning is the use of electronic media for a

variety of learning purposes that range from add-on functions in conventional classrooms to full substitution for the face-to-face meetings by online encounters”.

In order to support E-learning, the institutions usually employ a Learning Management System (LMS), which is a software application to assist, for example, in the administration, documentation, delivering of courses, applying tests, and tracking students' progress. In particular, LMS improves real-time interaction processes, allowing a student to have immediate contact with the teacher and other students.

There are many commercial and open source LMSs available, and most of them are Web-based for facilitating the users' access. In (E-learning, 2018) there are short descriptions of 302 LMSs, and two lists of the top 20 of these LMSs, one based on the best user experience, and the other based on the best customer experience. Moodle (Moodle, 2018) belongs to both lists and is employed by UFSCar to support the majority of its academic programmes. However, it is not used by the programmes based on active learning methodologies, such as Medicine and Nursing programmes, since this LMS is too generic for supporting the specificities of these programmes.

The UCG/UFSCar, which was established in 2002, has been developing projects focused mainly on the Education and Health areas. In the Health Education domain it developed, in partnership with DMed/UFSCar, the Electronic Reflective Portfolio (ERP) project (Forte, 2013). Recently, the UCG/UFSCar developed, in partnership with the DMed/UFSCar, the Teaching and Research Institute (TRI) of the Sírio-Libanês Hospital (SLH), and a software house, the LMS named Educational and Academic Management System for Courses Based on Active Learning Methodologies (EAMS-CBALM) (Santos, 2016).

2.4.1 Educational and Academic Management System for Courses Based on Active Learning Methodologies (EAMS-CBALM)

Scrum (Schwaber, 2017) was employed in the development of the EAMS-CBALM. Throughout the one-year development of this LMS project, weekly meetings were held at the software house in São Carlos-SP for the planning and analysis of its different phases, involving the UCG/UFSCar coordinator, the developer team coordinator of the software house, two POs designated by TRI/SLH, and a PhD's

student. There were also monthly meetings at TRI/SLH in São Paulo-SP to present the PIs resulting from the sprints, involving the participants of the software house meetings and TRI/SLH members directly or indirectly related to the project. In addition, the PhD's student participated on courses offered by the DMed/UFSCar and the TRI/SLH to observe the teaching-learning process of these courses.

During the software house meetings the POs reported stories, informally in Portuguese, describing what the activities of the EAMS-CBALM users would be. Then, from these stories system requirements were captured and specified also informally in Portuguese. Using these requirement specifications, the software house developer team defined system behaviour scenarios and implemented the screen prototypes for these scenarios, which were presented and discussed at the next meeting.

These meetings totalled 109 hours, which mainly allowed the understanding of CBALM teaching-learning process, defining the functional and non-functional system requirements, and defining the system architecture.

2.4.1.1 EAMS-CBALM Functional Architecture

The EAMS-CBALM has two main functional modules: academic and pedagogical. The academic module, illustrated in Figure 12, allows for registering the programme structure, the CBALM community, and the curriculum. The programme structure is used for the courses creation, where each course can be offered several times, and each course offer can have several classes. The CBALM community consists of students and teachers, and the latter can play the following roles: facilitator in curricular activities, supervisor, preceptor, consultant, appraiser, author of simulated practices and of educational units, and manager of educational units, working groups or educational resources. Registered students can be enrolled in courses, and then allocated in groups. The curriculum is used to create a curriculum structure with at least two and at most five hierarchical levels. For example, the UFSCar Medicine Programme, whose curriculum is illustrated in Figure 9, has five levels: cycles, years, educational units, curricular activities and educational actions.

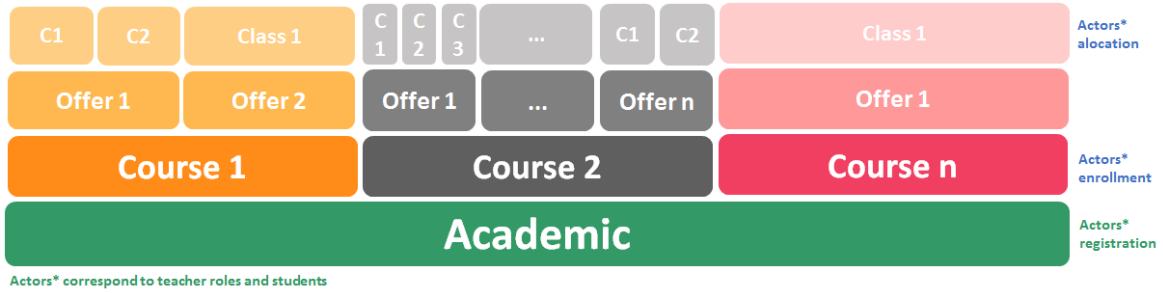


Figure 12: EAMS-CBALM academic module

The pedagogical module, illustrated in Figure 13, reflects the curricular structure over each class of each course previously recorded in the academic module. The educational actions are programmed in the pedagogical module giving rise to the educational environment on which students and teachers will perform and record their activities. The planning of an educational action starts with the meetings preparation, each meeting corresponding to a movement of the constructivist spiral, and ends with the evaluations preparation.

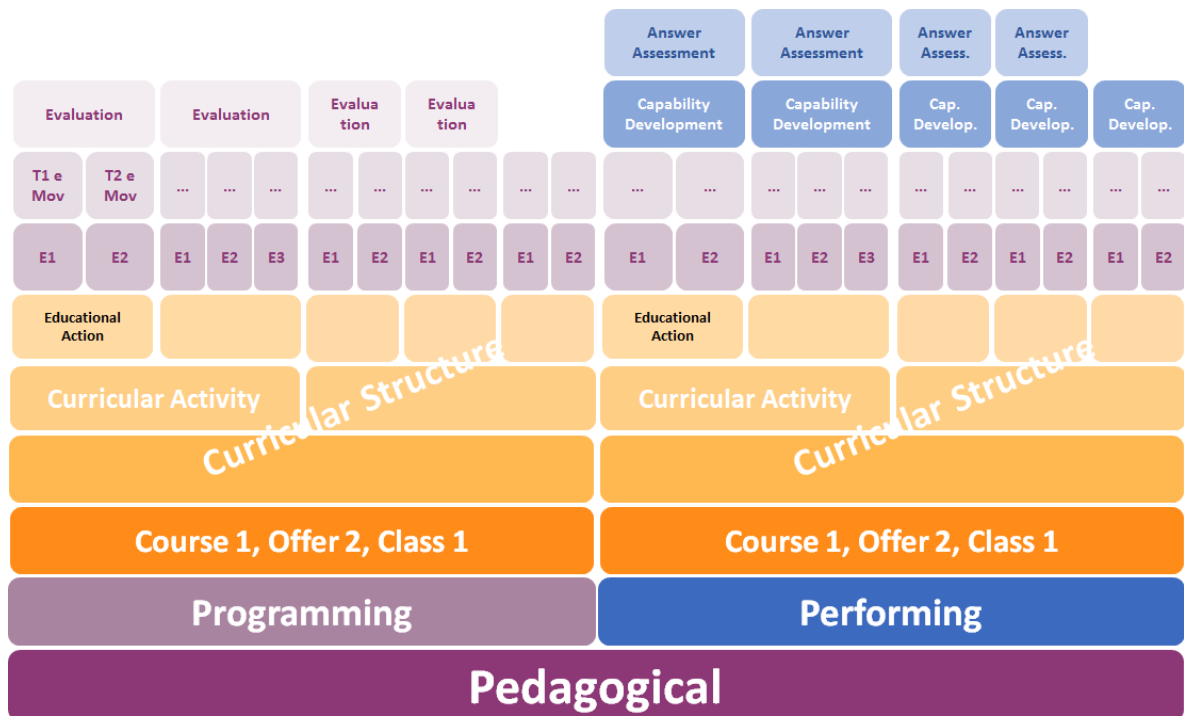


Figure 13: EAMS-CBALM pedagogical module

The EAMS-CBLAM allows for the discussion through forums, the sending of text documents, images, videos, and audios, the creation of individual or collaborative documents, the storage and presentation of document versions, and the

knowledge sharing produced by other students and teachers. All these resources contribute for the student capabilities development leading to the knowledge production.

Another EAMS–CBALM feature regards the data visualization, since all educational content produced during the teaching-learning process is provided by means of a curricular trajectory. This content includes from the teaching contribution until the individual and collaborative knowledge produced by the students.

The curricular trajectory can be viewed in a timeline or associated with an educational unit, allowing students and teachers to visualize all developed practices in the most adequately way at every moment. In addition, the produced knowledge is indexed by keywords, facilitating the search of any content created throughout the teaching-learning process.

2.4.1.2 EAMS-CBALM Implementation

In order to obtain scalability, interoperability, and maintainability the EAMS-CBALM was built decoupling the front-end from the back-end according to a Model–View–Controller (MVC) like pattern. This architectural pattern is illustrated in Figure 14: the Presentation layer is the front-end and corresponds to View; the Controller and Rest Services layer corresponds to Controller; and Model was split into a Business layer and an Integration and Persistence layer, to get more flexibility and to facilitate even more the maintenance of this system, and they compose the back-end.

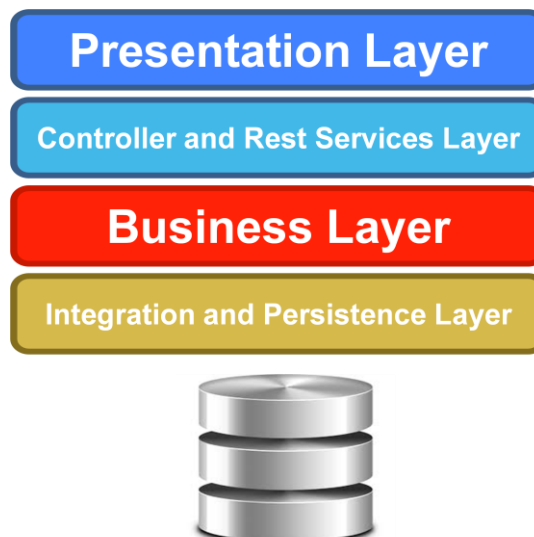


Figure 14: Architectural pattern of EAMS-CBALM implementation

The communication between Presentation and Business layers is done by controllers that are accessible through HTTP calls using REST services, allowing in this way the front-end to exchange documents in the JSON format with the back-end. The Presentation layer pages are not rendering on the back-end, the requests in the front-end take place through JavaScript calls encapsulated in the AngularJS framework, and this pages are rendered with their data through this framework. The Business layer is implemented through Grails Web framework services that encapsulate the processing logic, and access the Integration and Persistence layer, which is composed of domain classes. The EAMS-CBALM employs the PostgreSQL relational database.

By decoupling front-end and back-end, it is expected to reduce server burden, leaving it solely responsible for handling business rules and data persistence. In addition, this type of approach enables a more distributed deployment of the system, facilitating the load balancing and the resource scheduling on demand.

2.5 Final Considerations

In this chapter we presented a literature review on the main concepts and techniques that served as the basis for the development of this MSc's project. Throughout the chapter, the fundamental concepts discussed are related to agile software development, ontologies, active learning methodologies and learning management systems.

In the agile software development section, Scrum was presented and its roles, events and artefacts were briefly described. It was also pointed out, based on Dingsoyr research (Dingsoyr, 2016), that Scrum is, nowadays, the most relevant topic in agile software development practices among developes and researchers. Futhermore in this section, BDD was presented and its Ubiquitous language, templates, acceptance tests and implementation were briefly described. Then, studies that combined Scrum with BDD were shown and discussed.

In the ontologies section, the concepts, classifications, main ontologies languages and tools were presented and discussed. Inside that context, we selected OWL and Protégé as the ontology language and tool for this MSc's project. Furthermore, both were illustrated using HERO, a domain ontology created by Zemmouchi-Ghomari (Zemmouchi-Ghomari, 2013), that is also used in this project to develop the main ontologies presented in the next chapters.

In the active learning methodologies section, we presented the UFSCar Medicine Programme, a that follows a socio-constructivist educational approach, has a competency-oriented pedagogical programme, and employs active learning methodologies, such as PBL (Rhem, 1998), and Practice Based Learning (Carlisle, 2009). Details were given about the Programme structure and its educational activities to provide a clear overview of how this formal education works.

In order to provide computational support for Programmes that employ active learning methodologies such as the UFSCar Medicine Programme, the learning management systems section describes the EAMS-CBALM software (Santos, 2016). This software was developed by the the Ubiquitous Computing Group (UCG) of UFSCar in partnership with the UFSCar Medicine Department, the Teaching and Research Institute of the Sírio-Libanês Hospital, and a local software house. It was based on this software development experience and the literature review, that this MSc's project was conceived and developed as shown in the following chapter.

Chapter 3

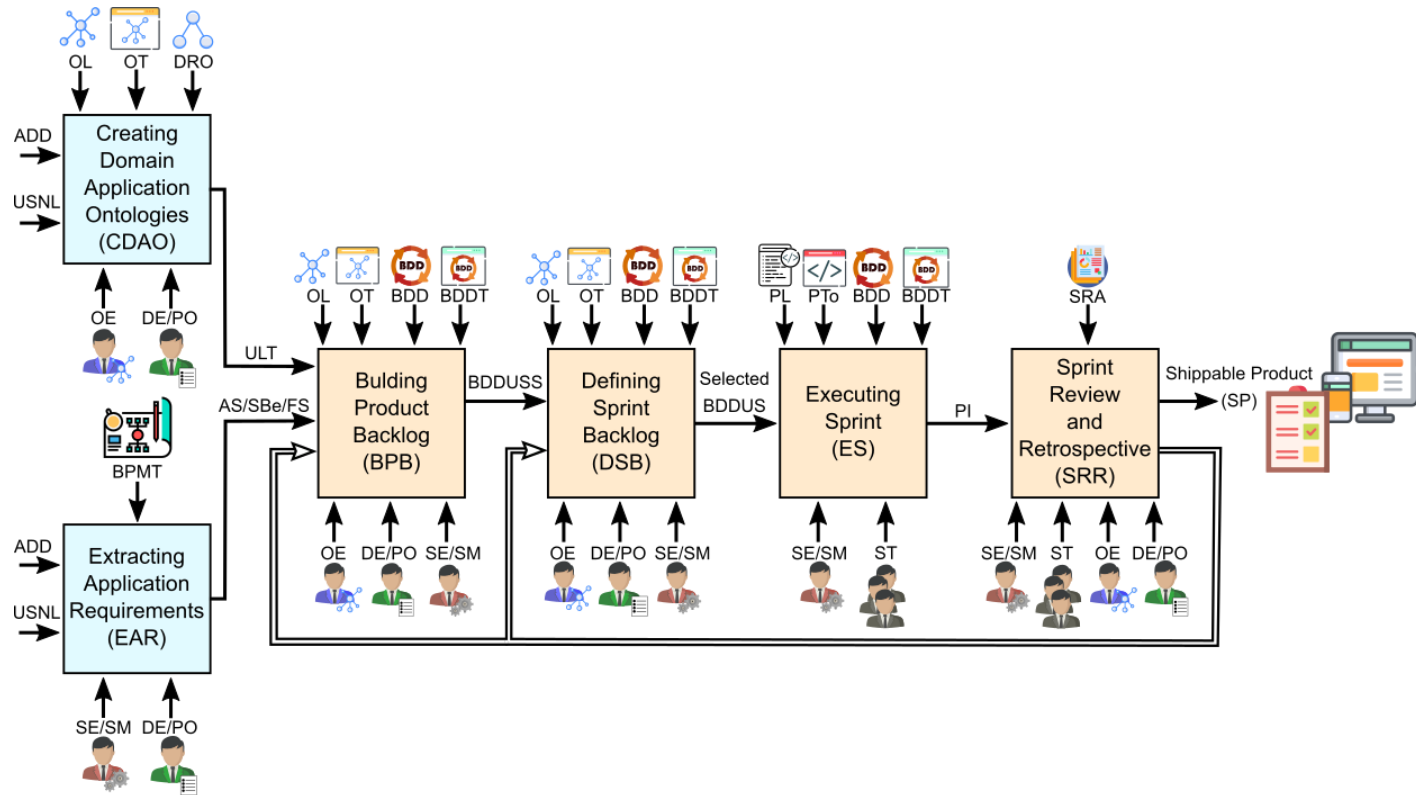
SCRUMONTOBDD

This chapter presents the ScrumOntoBDD approach for agile software development, which combines Scrum, Ontology and BDD. Section 3.1 gives an overview of this approach focusing on its main activities, and from Section 3.2 until Section 3.7 these activities are detailed and illustrated with excerpts of the case studies related to the EAMS-CBLAM development; Section 3.8 has the final considerations.

3.1 Overview

Figure 15 provides an overview of the ScrumOntoBDD approach by means of a SADT-like diagram, where a rectangle represents a main activity of this approach, and an arrow on a rectangle can have one of the following meanings: on the left-hand side an activity input; on the right-hand side an activity output; at the top a resource employed in the activity; and at the bottom represents a role of the Development Team (DT).

Chapter 3: ScrumOntoBDD



LEGENDA:

→ ADD - Application Domain Documents	OE - Ontology Engineer	OL - Ontology Languages	BDD - Behaviour-Driven Development
→ USNL - User Stories in Natural Language	DE - Domain Expert PO - Product Owner	OT - Ontology Tools	BDDT - BDD Tools
→ ULT - Ubiquitous Language Terminology	SE - Software Engineer SM - Scrum Master	DRO - Domain Reference Ontologies	PL - Programming Languages
→ BDDUSS - BDD User Stories and Scenarios	ST - Scrum Team	BPMT - Business Process Modeling Techniques	SRA - Sprint Records and Annotations
→ PI - Product Increment			
AS - Application Scenarios			
SBe - System Behaviours			
FS - Feature Sets			

Figure 15: ScrumOntoBDD overview

The main ScrumOntoBDD activities are: *Creating Domain Application Ontologies (CDAO)*, *Extracting Application Requirements (EAR)*, *Building Product Backlog (BPB)*; *Defining Sprint Backlog (DSB)*, *Executing Sprint (ES)*, and *Sprint Review and Retrospective (SRR)*. CDAO and EAR can be performed in parallel with some overlapping. ES can be performed several times until a given PI is achieved, and after SRR is performed and the PI is accepted, a new sprint backlog is defined. The sprint review and sprint retrospective were joined in the same activity since they are performed in sequence involving all the DT members. The *Shippable Product (SP)* will be complete after all PB items have been processed successfully.

The inputs and outputs of these activities are: *Application Domain Documents (ADD)*, *User Stories in Natural Language (USNL)*, *Ubiquitous Language Terminology (ULT)*, *Application Scenarios (AS)*, *System Behaviours (SBe)*, *Feature Sets (FS)*, *BDD User Stories and Scenarios (BDDUSS)*, *Selected BDDUSS*, and *Product Increment (PI)*.

The main resources employed by these activities are: *Ontology Languages (OL)*, *Ontology Tools (OT)*, *Domain Reference Ontologies (DRO)*, *Business Process Modeling Techniques (BPMT)*, *BDD*, *BDD Tools (BDDT)*, *Programming Languages (PL)*, *Programming Tools (PTo)*, and *Sprint Records and Annotations (SRA)*.

The DT should be multidisciplinary and must accomplish the following roles:

(a) *Domain Expert (DE)* is responsible for sharing his/her knowledge of the application domain, and for transferring the necessary information to create the ontologies and to extract the application requirements;

(b) *Ontology Engineer (OE)* is responsible for creating, manipulating and evaluating the domain application ontologies, and also for assisting the formalization of knowledge and of software requirements;

(c) *Software Engineer (SE)* is responsible for tracking the entire software life cycle, ensuring its building and proper working according to the application requirements, and for contributing to the generation of quality artefacts; and

(d) *Product Owner (PO)*, *Scrum Master (SM)*, and *Scrum Team (ST)* have the same Scrum roles described in section 2.1.1 of this document, and additionally SM and ST must also be able to handle BDD and BDD tools.

Some DT roles can be played by the same person. For example, the PO designated by the customer could be a DE, and the SM designated by the software house could be a SE.

These main activities of ScrumOntoBDD will be detailed in the next sections, and will be illustrated using excerpts of the case studies reported in (Souza, 2017) and (Souza, 2018) related to the EAMS-CBALM development.

3.2 Creating Domain Application Ontologies (CDAO)

The CDAO activity can start from a reference ontology for the application domain, and if there is no such ontology it can be built. Then this reference ontology must be gradually specialized to achieve an ontology containing the main terminology of the ubiquitous language to be employed in the application development. Table 3 summarizes the inputs, outputs, resources, roles and the main tasks of this activity.

Inputs	Application Domain Documents (ADD) User Stories in Natural Language (USNL)
Outputs	Ubiquitous Language Terminology (ULT)
Resources	Ontology Languages (OL) Ontology Tools (OT) Domain Reference Ontologies (DRO)
Roles	Ontology Engineer (OE) Domain Expert (DE) Product Owner (PO)
Tasks	(a) Search ontologies to select or create a DRO for the application domain; (b) Successively specialize DRO based on application domain characteristics; (c) Validate the created domain application ontologies; and (d) Extract the main ULT from these ontologies.

Table 3: Inputs, outputs, resources, roles and tasks of CDAO activity

3.2.1 Ubiquitous Language Terminology for EAMS-CBALM

Based on HERO key concepts shown in Figure 7, and according to the higher education areas described by the National Council for Scientific and Technological Development (CNPq) of Brazil, we extended this reference ontology to be able to describe the Brazilian universities. Figure 16 shows this extension with the CNPq major educational areas.

Founded in 1968, UFSCar is a Brazilian public higher education institution, headquartered in the city of São Carlos, it has forty-eight academic departments, located in eight centers distributed over four campi, and its academic programmes are organized according to the CNPq large education areas.

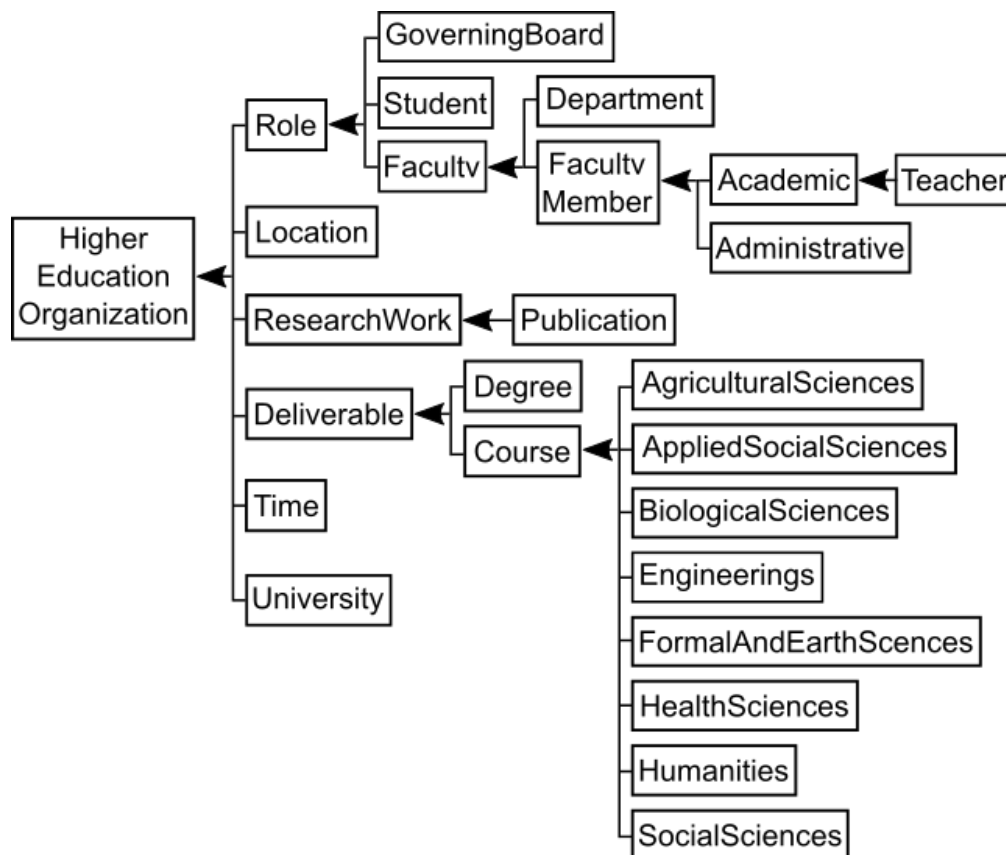


Figure 16: HERO key concepts with CNPq major education areas

We applied the concepts shown in Figure 16 to define using Protégé 5.2 (Horridge, 2007) an UFSCar ontology that represents all the programmes offered by this university. Figure 17 depicts an excerpt of this ontology in OWLViz, which shows

the eight CNPq large educational areas, and focuses on the ten UFSCar programmes in the Health Sciences area, where the Medicine Programme is located.

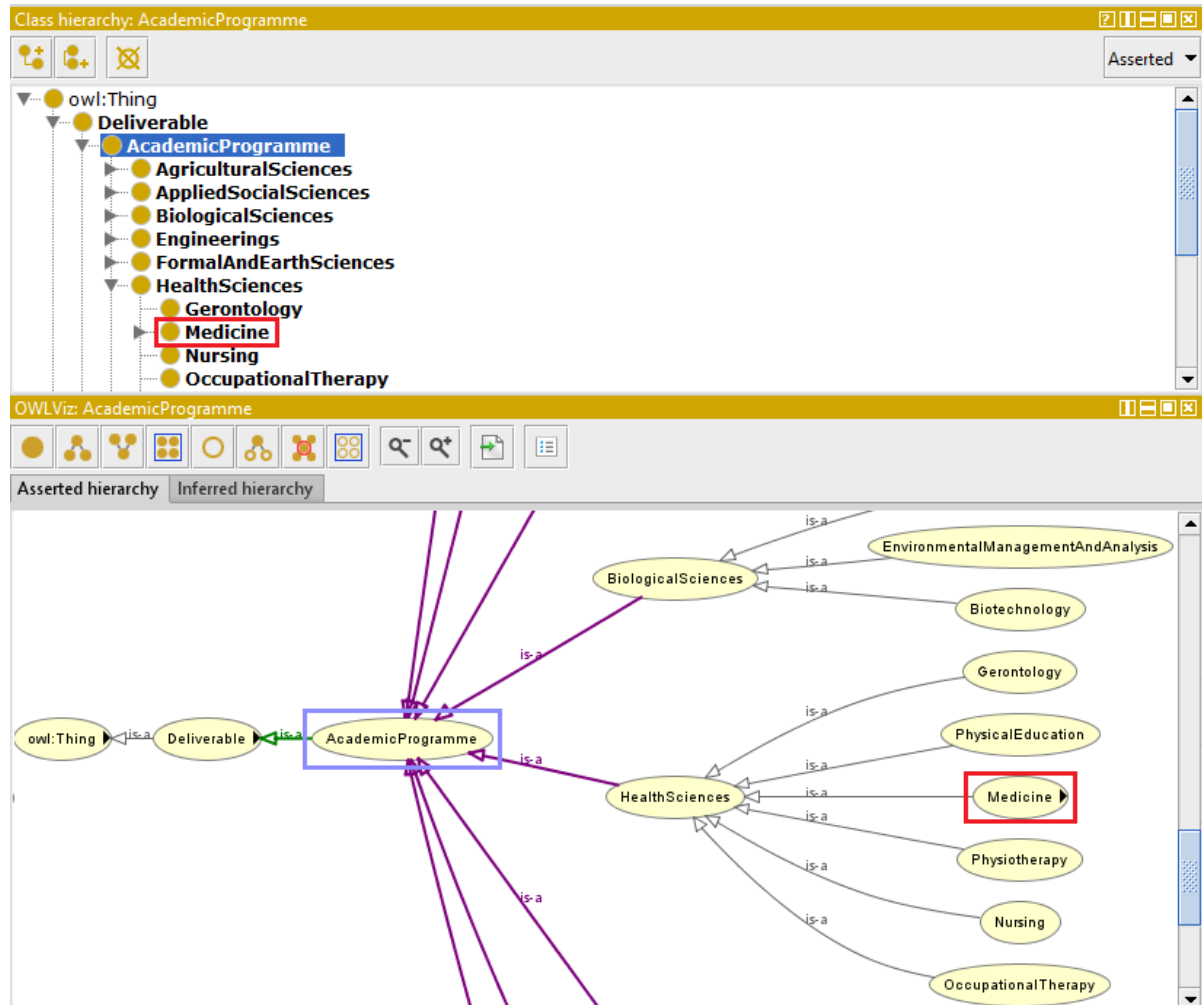


Figure 17: UFSCar ontology in OWLViz

Using the UFSCar Medicine Programme curriculum described in (UFSCar, 2007), we specialized the UFSCar ontology for obtaining the UFSCar Medicine Programme ontology, which contains the main ULT to be employed in SGPA-CBMAA development. Figure 18 shows an excerpt of the UFSCar Medicine Programme ontology, focusing on its programme structure, and Figure 19 shows another excerpt of the same ontology, the asserted class hierarchy view focusing on its learning methodology, some of its educational activities, and the steps of the constructivist spiral.

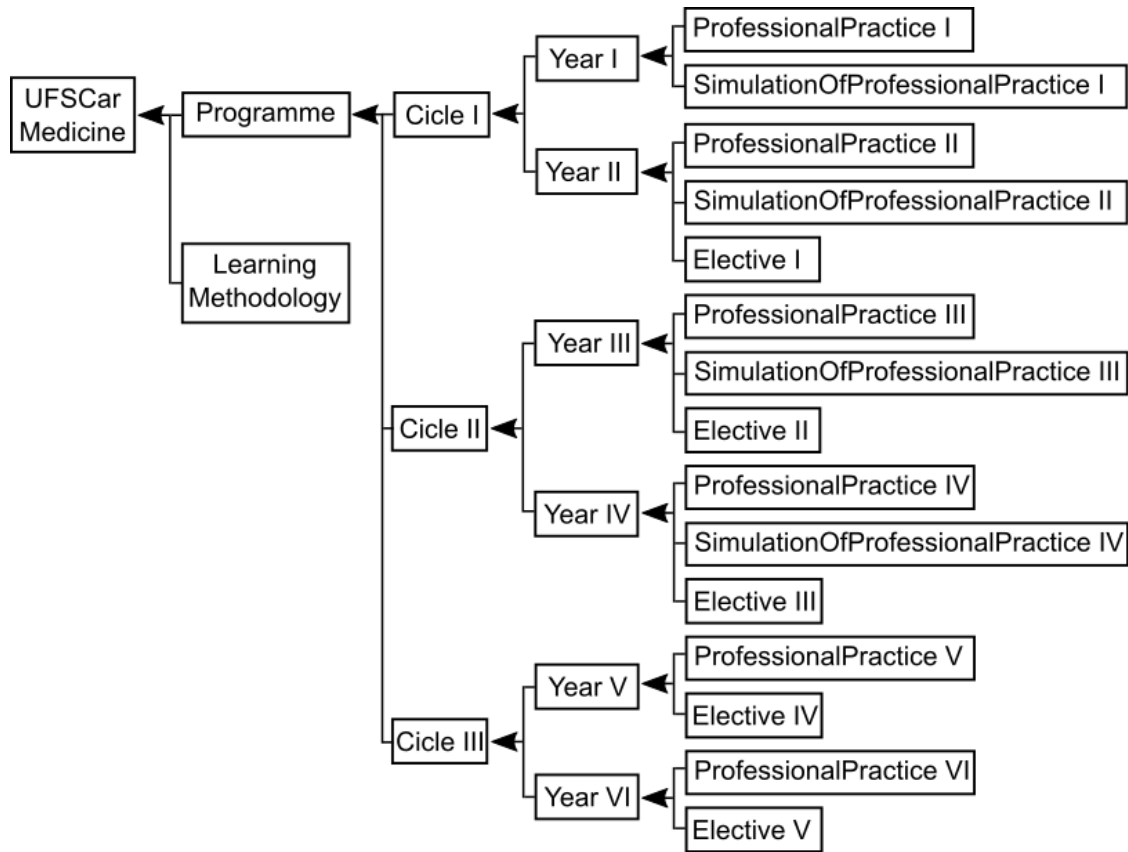


Figure 18: UFSCar Medicine Programme ontology

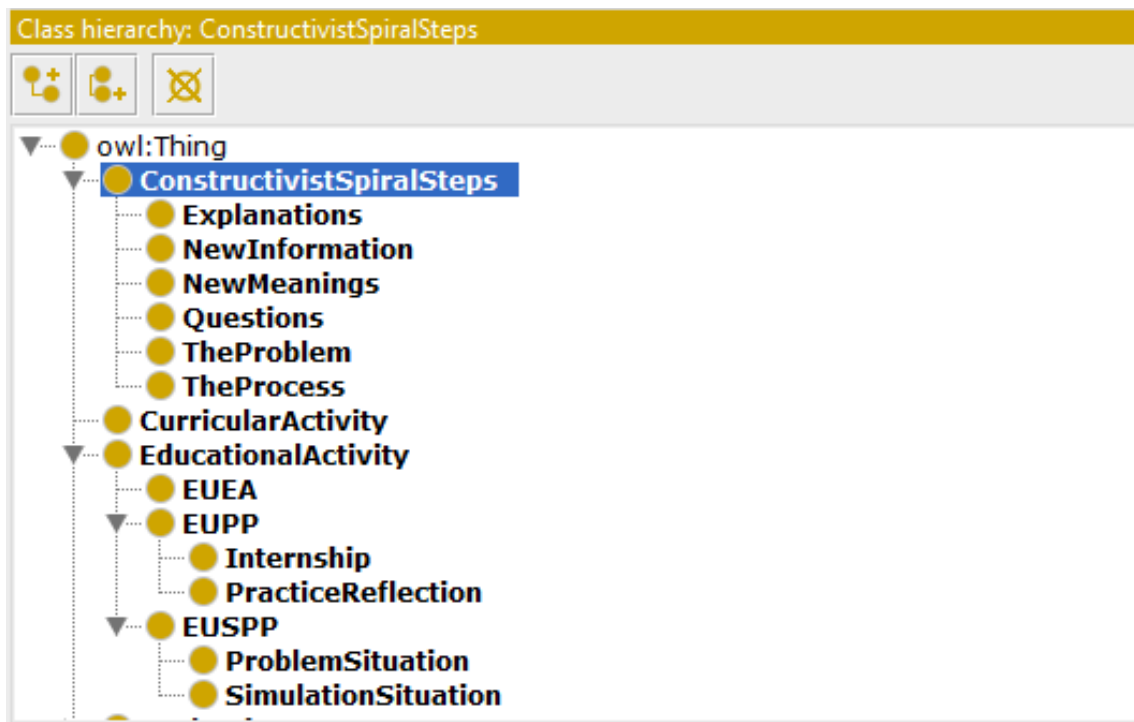


Figure 19: Constructivist spiral steps and educational activities of UFSCar Medicine Programme ontology

The Evaluation Management module of EAMS-CBALM generated the most controversies between PO and developers. This module is composed of fourteen functional requirements, and even the scenarios of Evaluation Instrument Register requirement having been redone several times, its final implementation did not fully satisfy the PO. Therefore, we selected this module and this requirement for continuing the illustration ScrumOntoBDD.

3.2.1.1 Ubiquitous Language Terminology for Evaluation Management Module

The student evaluation process of the UFSCar Medicine Programme is different and more complex than the traditional one. According to UFSCar (UFSCar, 2007), evaluations are performed by all people involved in the educational activities, expressing their perceptions, indicating the relevant aspects, and aspects that need to be improved, reworked or replaced. There are two evaluation types in this process, namely formative and summative evaluation, and the possible results to be received/given are 'satisfactory', 'needs improvement' and 'unsatisfactory'. The student who does not reach a satisfactory result must undertake an improvement plan proposed by the teacher, and then be re-evaluated. Evaluations are consolidated by applying the following set of six instruments types:

(a) Performance Assessment of the Teaching-Learning Process (PATLP): It is formative. Teacher evaluates student. The Respondent is the teacher. A PATLP has three steps: teacher evaluation, classmate evaluation, and improvement plan if necessary;

(b) Reflective Portfolio (RP): It is formative and summative. The monitoring of each student's portfolio by the teacher is part of the formative assessments. The summative evaluation refers to its preparation and presentation according to pre-established delivery dates that must be recorded into the system;

(c) Written Examination (WE): It is summative. Each WE consists of questions created by the teacher, answered by the student, and then assigned by the teacher. Each question is individually analysed in order to determine the student progress. All question must have a 'satisfactory' result for the student to pass the WE. Questions

that failed to get a 'satisfactory' result are considered as progress deficit and will be worked out in the next WE;

(d) Progress Test (PT): It is formative and summative. A PT consists of multiple choice questions. The teacher monitors the performance of each student's performance in a PT as part of the formative assessments. A PT is also summative because the students' presence at a PT gives them a 'satisfactory' result;

(e) Objective and Structured Evaluation of Professional Performance (OSEPP): It is summative. Instead of questions like WE, the students act in clinical cases and are evaluated by the teacher. The OSEPP evaluation is similar to the WE evaluation; and

(f) Problems Based Exercise (PBE): It is formative. A PBE assesses the student's individual ability to study and identify health needs, formulate patient and family problems, and propose a healthcare plan for a particular context and problem situation.

Table 4 shows the acronyms employed by these evaluation instruments types, and the relationships between the involved actors.

Instrument type	Respondent	Appraiser	Appraisee
PATLP	Teacher	Teacher	Student
RP/PT	Student	Teacher	Student
WE/OSEPP/PBE	Student	Teacher	Student

Table 4: Evaluation instruments types

Based on this information and on Gangemi (Gangemi, 2006), we defined the terminology (names, adjectives, and verbs) to be formally represented in the Evaluation Process ontology for the UFSCar Medicine Programme in terms of concepts, attributes and relations. Figure 20 shows an excerpt of this ontology. Classes *EducationalActivity* and *EvaluationProcess* model the concept of *CurricularActivity* in the Programme. Each *EducationalActivity* starts with one or more meetings. A *Meeting* has the participation of students and teachers, each one with specific *Roles*, and it also triggers a *LearningTrigger*. Each *LearningTrigger*

transverse the *ConstructivistSpiralSteps* and ends with an *EvaluationProcess*. *EvaluationProcess* is consolidated by applying *EvaluationInstruments*.

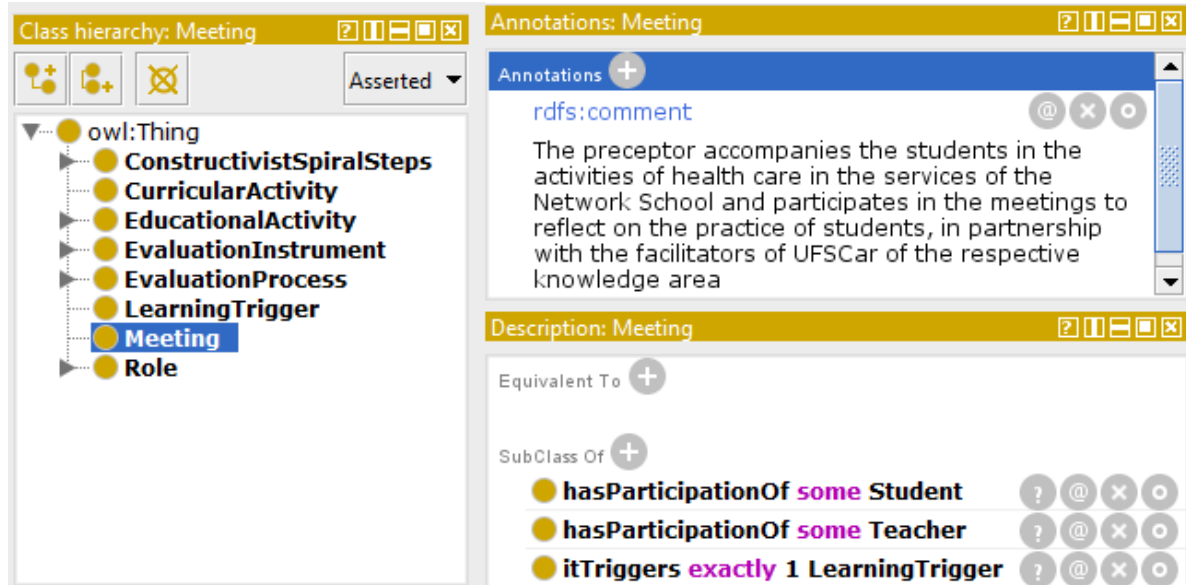


Figure 20: Evaluation Process ontology root classes (left-side) and *Meeting* class relations (right-side)

Figure 21 shows an excerpt of the Evaluation Process ontology of the UFSCar Medicine Programme, focusing on its evaluation instruments types, and showing the PATLP instrument and the relationships between their actors.

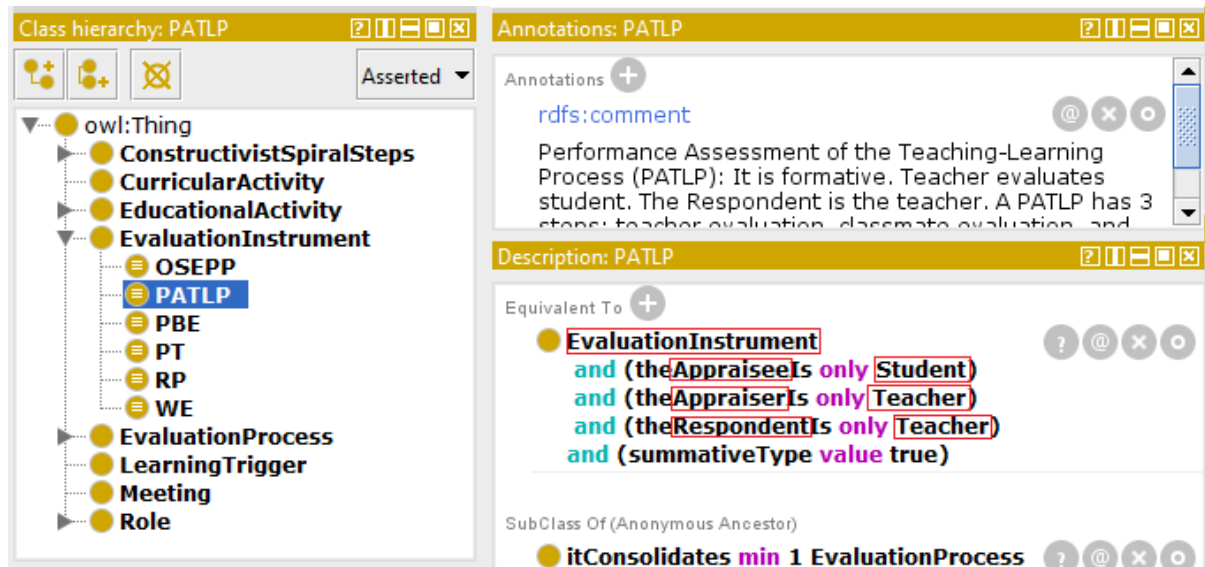


Figure 21: Evaluation Process ontology with PATLP

3.2.2 Ontology Validation of EAMS-CBALM

We assessed the quality of our ontologies according to the structural and functional dimensions (Gangemi, 2006). Structural validation considers the ontology logical structure, focusing on its syntax and formal semantics. Protégé offers several ontology verification tools for detecting inconsistencies and redundancies in ontologies. For all ontologies we used the reasoners FaCT++, HermiT and Pellet, and we discovered neither inconsistencies nor redundancies on them.

An example of this structural validation is shown in Figure 22, where FaCT++, Hermit and Pellet have succeeded processing the Evaluation Process ontology in 890, 2125, and 568 milliseconds respectively. The current version of this ontology bears an amount of 358 axioms, 182 of them being logical axioms, 42 classes, 24 object properties, 9 data properties, and 0 individuals.

```

INFO 15:43:29 ----- Running Reasoner -----
INFO 15:43:29 Pre-computing inferences:
INFO 15:43:29   - class hierarchy
INFO 15:43:29   - object property hierarchy
INFO 15:43:29   - data property hierarchy
INFO 15:43:29   - class assertions
INFO 15:43:29   - object property assertions
INFO 15:43:29   - data property assertions
INFO 15:43:29   - same individuals
INFO 15:43:30 Ontologies processed in 890 ms by FaCT++
INFO 15:43:59 ----- Running Reasoner -----
INFO 15:43:59 Pre-computing inferences:
INFO 15:43:59   - class hierarchy
INFO 15:43:59   - object property hierarchy
INFO 15:43:59   - data property hierarchy
INFO 15:43:59   - class assertions
INFO 15:43:59   - object property assertions
INFO 15:43:59   - data property assertions
INFO 15:43:59   - same individuals
INFO 15:44:02 Ontologies processed in 2125 ms by null
INFO 15:44:14 ----- Running Reasoner -----
INFO 15:44:14 Pre-computing inferences:
INFO 15:44:14   - class hierarchy
INFO 15:44:14   - object property hierarchy
INFO 15:44:14   - data property hierarchy
INFO 15:44:14   - class assertions
INFO 15:44:14   - object property assertions
INFO 15:44:14   - data property assertions
INFO 15:44:14   - same individuals
INFO 15:44:14 Ontologies processed in 568 ms by Pellet

```

Figure 22: Results of Evaluation Process ontology processing: FaCT++ (top), HermiT (middle) and Pellet (bottom)

Functional assessment focused on the ontologies usage. This assessment includes evaluation by domain experts, user satisfaction, task assessment, and topic assessment. Our ontologies were assessed in collaboration with our PO during development, aiming at verifying their structure, their restrictions, relations between concepts and the attributes of their concepts. The PO assessed how well these ontologies met predefined criteria, and they have been updated according to the PO suggestions.

3.3 Extracting Application Requirements (EAR)

The EAR activity starts by identifying the customer needs, the objective, scope and scenarios of the application. Then these scenarios are described by means of Business Process Modeling Techniques (BPMT) (TilyFy, 2018), such as UML diagrams, Business Process Model Notation (BPMN) (OMG, 2011) and flowcharts, to illustrate key application requirements. Based on this description, the main functionalities of the software are defined, and the system behaviours and features for providing such functionalities are described. Table 5 summarizes the inputs, outputs, resources, roles and the main tasks of this activity.

Inputs	Application Domain Documents (ADD) User Stories in Natural Language (USNL)
Outputs	Application Scenarios (AS) Feature Sets (FS) System Behaviours (SBe)
Resources	Business Process Modeling Techniques (BPMT)
Roles	Software Engineer (SE) Scrum Master (SM) Domain Expert (DE) Product Owner (PO)
Tasks	(a) Identify customer needs and application characteristics; (b) Draw up AS; (c) Define the main software functionalities; and (d) Draw up FS and SBe.

Table 5: Inputs, outputs, resources, roles and tasks of EAR activity

3.3.1 Requirements for EAMS-CBALM

To continue illustrating ScrumOntoBDD, we choose the *Instrument Types Registration* requirement that was extracted from the following user story reported by the PO during the EAMS-CBALM development:

“In order for the programme coordinator to carry out a student evaluation, the six instrument types have to be previously registered. This requirement is needed because the evaluation form heading changes according to the employed instrument. When registering an instrument, the system must keep the following information: name, acronym and the relationship between who responds to the evaluation, who evaluates and who is evaluated. This last information is crucial since in conjunction with the curricular activity it defines which form type is applied when registering an evaluation.”

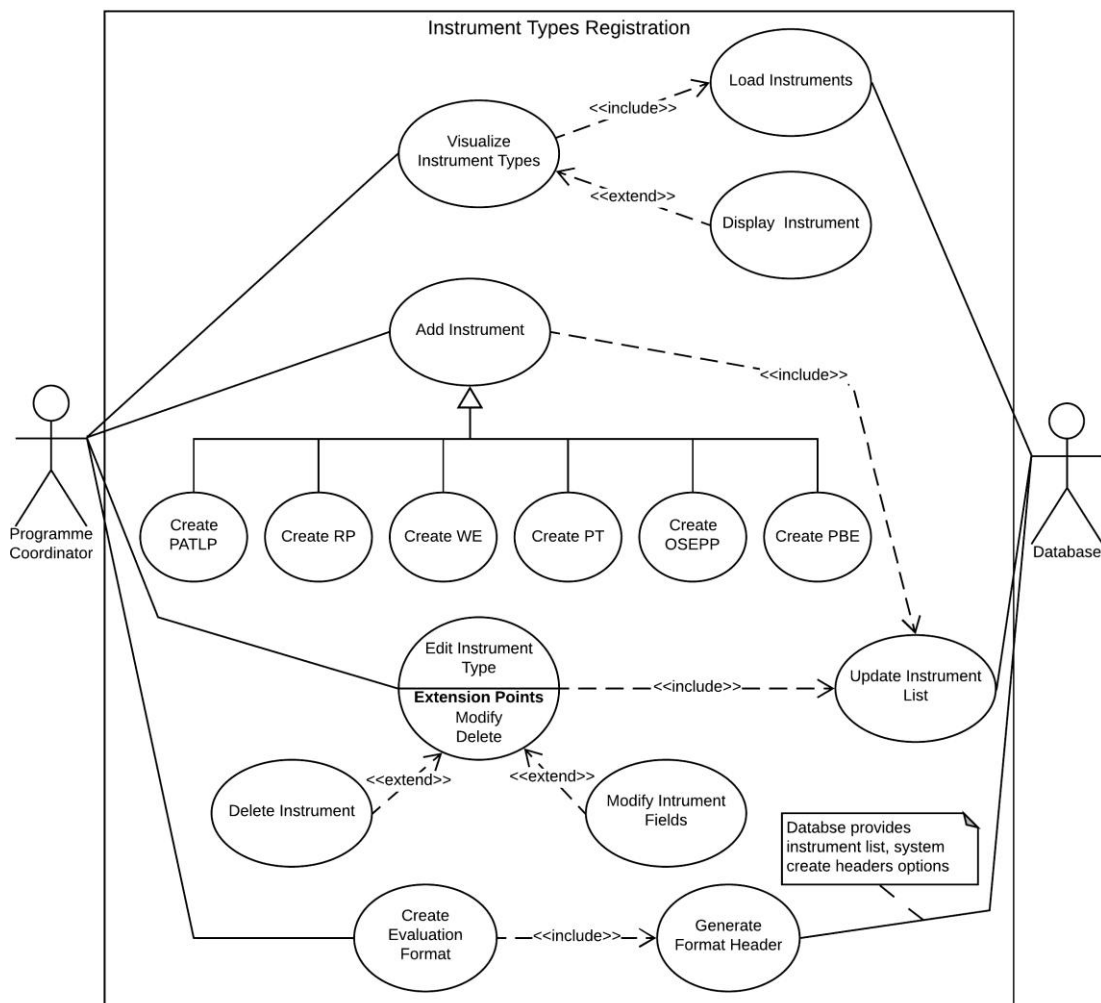


Figure 23: UML use case diagram for Instrument Types Registration requirement

Based on this user story and the Instrument Types Registration requirement, we identified the application scenarios by means the UML use case diagram showed in Figure 23.

Feature: add instrument type			
In order to carry out a student evaluation			
As a programme coordinator			
I want to register the instrument types by filling all necessary information into the system			
Background:			
Given the following instrument types exist			
Name		Acronym	
Performance Assessment of the Teaching-Learning Process		PATLP	
Reflective Portfolio		RP	
Written Examination		WE	
Progress Test		PT	
Objective and Structured Evaluation of Professional Performance		OSEPP	
Problems Based Exercise		PBE	
And the following actors and roles exist			
Actor	Role 1	Role 2	
Student	Respondent	Appraisee	
Teacher	Respondent	Appraiser	
And the following relationship exist			
Instrument	Respondent	Appraiser	Appraisee
PATLP	Teacher	Teacher	Student
RP / PT / WE / OSEPP / PBE	Student	Teacher	Student
Behaviour 1: Register a PATLP into the system			
Given the name is Performance Assessment of the Teaching-Learning Process			
And the acronym is PATLP			
And the respondent is a student			
And the appraiser is a teacher			
And the appraisee is a student			
When the user clicks in Register			
Then the system register all the instrument fields that were fulfilled			
And the PATLP form header for student evaluation is created			
And the Evaluation Instrument list is updated			
Behaviour 2: [...]			

Figure 24: System feature set for Instrument Types Registration requirement

Based on these application scenarios, we identified the following feature set: *Visualize Instrument Type*, *Add Instrument*, *Edit Instrument Type*, and *Create Evaluation Format*. Figure 23 illustrates the *Add Instrument* feature with one of its expected system behaviour using a BDD-like notation.

3.4 Building Product Backlog (BPB)

The BPB activity starts by defining the user stories and scenarios that will compose the PB items, using the ubiquitous language terminology, the application scenarios, and the system behaviours and features. Then these items must be formally specified, by means of an ontology model that employs BDD templates, and listed in a priority order of implementation, building in this way a formally described PB. Table 6 summarizes the inputs, outputs, resources, roles and the main tasks of this activity.

The BDD scenario template is alike to an Extended Finite State Machine (EFSM) model (El-Fakih, 2016), and was formally defined through Protégé OWL ontology in (Silva, 2017). An excerpt of this ontology is shown in Figure 25.

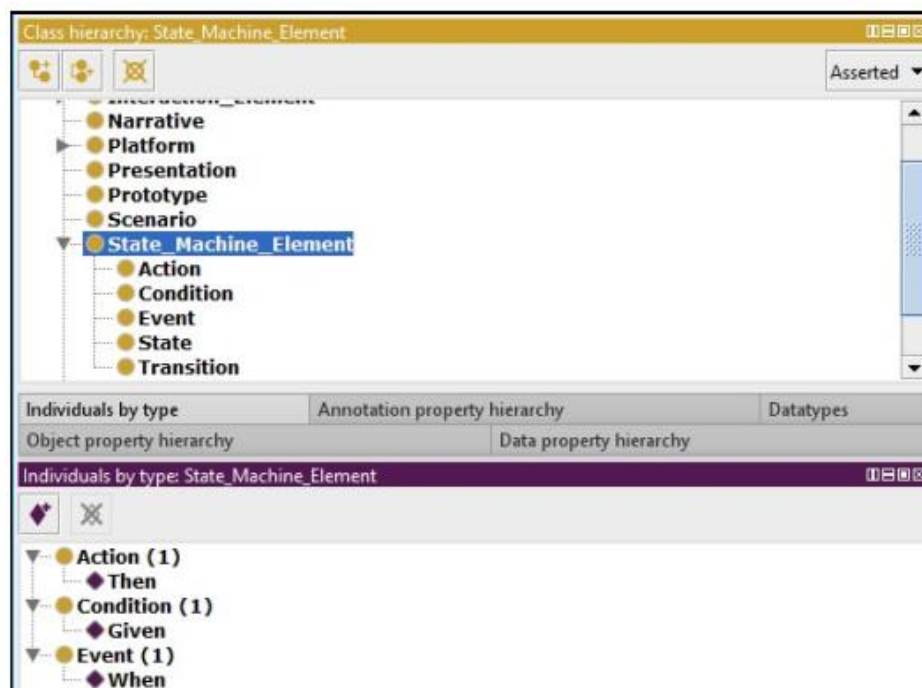


Figure 25: OWL ontology for the BDD scenario template (from (Silva, 2017))

This ontology was employed for describing concepts used by platforms, models and artefacts that compose the design of interactive systems, allowing a wide description of User Interface (UI) elements and its behaviors in order to specify scenarios and to support acceptance testing. A scenario meant to be run in a given UI is represented as a transition, and states are used to represent the original and resulting UIs after a transition occurs. Scenarios in the transition have at least one or more *conditions*, one or more *events*, and one or more *actions*, which are represented by the *given*, *when* and *then* clauses respectively.

In order to formally specify the user stories of the PB using ontologies and BDD templates, a User Story and Scenario ontology can be built employing a similar EFSM model.

Inputs	Ubiquitous Language Terminology (ULT) Application Scenarios (AS) Feature Sets (FS) System Behaviours (SBe)
Outputs	BDD User Stories and Scenarios (BDDUSS)
Resources	Ontology Languages (OL) Ontology Tools (OT) BDD BDD Tools (BDDT)
Roles	Ontology Engineer (OE) Domain Expert (DE) Product Owner (PO) Software Engineer (SE) Scrum Master (SM)
Tasks	(a) Define User Stories and Scenarios (USS) using ULT, AS, FS, and SBe; (b) Formally specify the defined USS by means of a User Story and Scenario ontology that employs BDD templates; and (c) List the specified BDDUSS in a priority order thus building the Product Backlog.

Table 6: Inputs, outputs, resources, roles and tasks of BPB activity

3.4.1 User Story and Scenario Ontology for EAMS-CBALM

Although BDD user stories and scenarios follow the templates described in North (North, 2007) and illustrated in Figure 4, BDD tools generally do not strictly

follow these models. For example, JBehave (JBehave, 2015) supports a slightly different user story template and the same scenario template, which are shown in Figure 26.

<p>Narrative: <i>[story title]</i> In order to <i>[benefit]</i> As a <i>[role]</i> I want to <i>[feature]</i></p>	<p>Scenario: <i>[scenario title]</i> Given <i>[main context]</i> And <i>[additional contexts]</i> When <i>[specific event]</i> Then <i>[main outcome]</i> And <i>[additional outcome]</i></p>
--	--

Figure 26: JBehave User Story and Scenario templates

There are six scenarios in our user story example, one for each instrument type. Figure 27 illustrates the EFSM model we employed for building the User Story and Scenario ontology for EAMS-CBALM. This example deals with the registering of the PATLP instrument type.

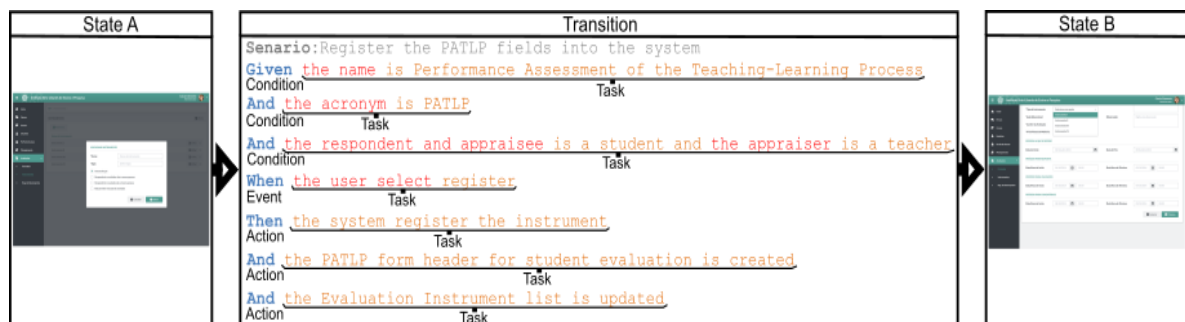


Figure 27: EFSM model representing PATLP instrument type registration

The ontology presented in (Silva, 2017) is domain-free, since it only describes behaviors that report steps of scenarios performing actions on UIs through interaction elements. Therefore, specific business behaviors of our user story example had to be specified, since most of them employ the domain terminology extracted from the Evaluation Process ontology. Hence, we had to map each terminology to a user interaction (e.g., click, selection), and to write steps for each one of these interactions. Figure 28 shows an excerpt of the User Story and Scenario ontology for EAMS-CBALM dealing with the registration of the PATLP instrument type.

Figure 28: User Story and Scenario ontology with PATLP instrument type registration

3.5 Defining Sprint Backlog (DSB)

The DSB activity starts with a sprint planning meeting for defining which PB items will be implemented on that sprint. Then the tasks to be performed for each one of these items must be described, and the work to accomplish them must be planned. Table 7 summarizes the inputs, outputs, resources, roles and the main tasks of this activity.

Inputs	BDD User Stories and Scenarios (BDDUSS)
Outputs	Selected BDD User Stories and Scenarios (Selected BDDUSS)
Resources	Ontology Languages (OL) Ontology Tools (OT) BDD BDD Tools (BDDT)
Roles	Ontology Engineer (OE) Domain Expert (DE) Product Owner (PO) Software Engineer (SE) Scrum Master (SM)
Tasks	(a) Select the BDDUSS of the Backlog to be implemented; and (b) Define the tasks and work for implementing the Selected BDDUSS.

Table 7: Inputs, outputs, resources, roles and tasks of DSB activity

3.5.1 Tasks for Implementing the Selected BDDUSS of EAMS-CBALM

When employing BDD, one the task to be performed in the ES activity is the generation of the acceptance tests before implementing the selected BDDUSS itself. Since it is possible to reuse the steps of the User Story and Scenario ontology in multiple testing scenarios, one task that can be done in DSB activity is to write the scenarios for acceptance testing that will be applied later in ES activity, avoiding in this way misinterpretations of feature sets during that activity.

Nevertheless, it is necessary first to map the user stories and scenarios described in OL to BDDT language. Depending on the used OT and BDDT, there may be tools available that gives support to this mapping. In this case study we made a script that starts by reading the OWL file, describing the User Story and Scenario ontology, and at the same time selects all BDD scenarios and its corresponding steps. Then, a JBehave textual story file is created, and filled with the selected information using the JBehave scenario template. This textual story file will be used later to structure all acceptance tests related to the scenarios containing in this file. Figure 29 shows an excerpt of the User Story and Scenarios OWL file transcript to JBehave textual story file.

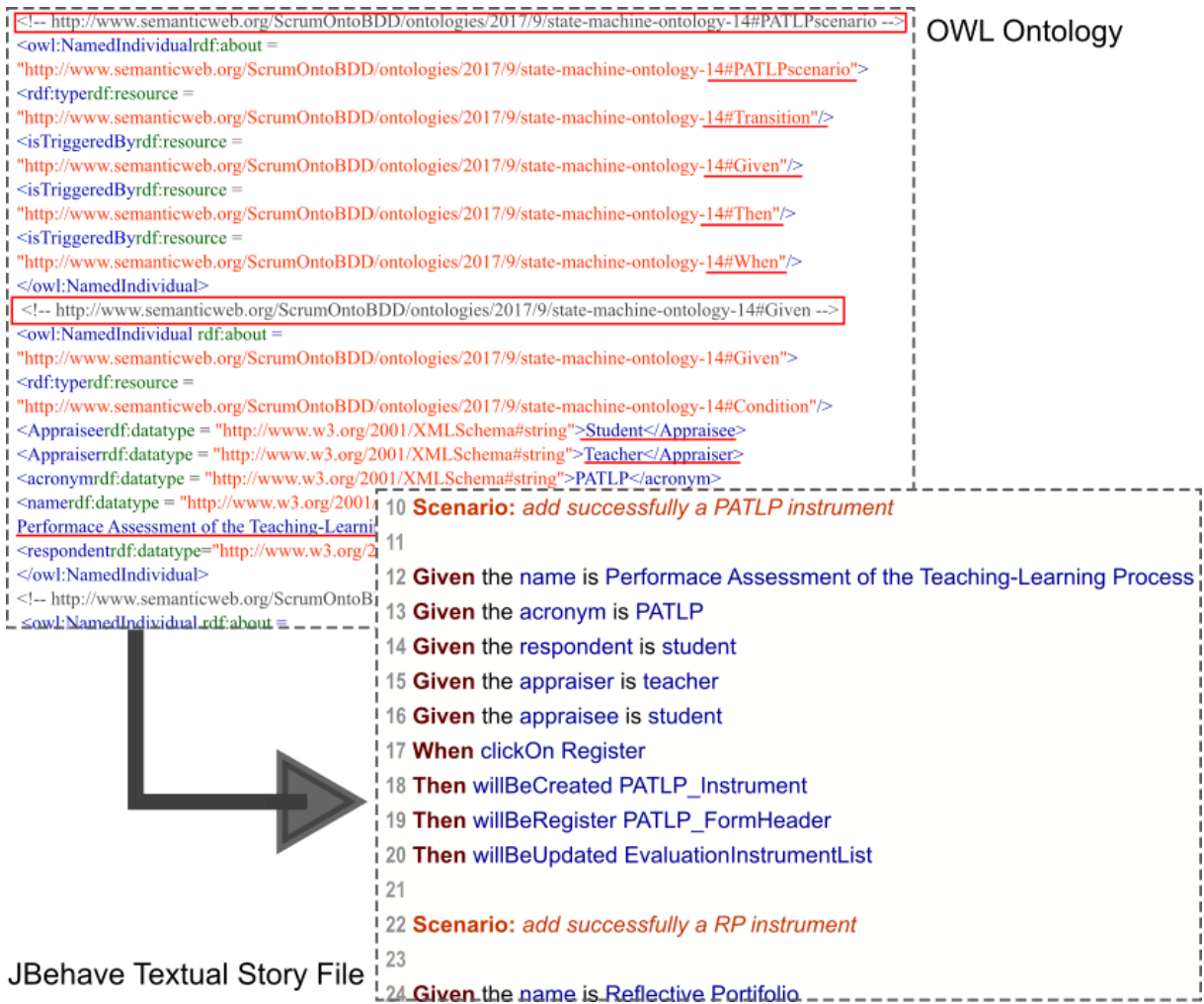


Figure 29: User Story and Scenario ontology transcript to JBehave textual story file

3.6 Executing Sprint (ES)

The ES activity starts with a daily meeting for analyzing the work done until now and planning the work to be done on that day to implement the selected user stories and scenarios. The daily meeting is held every day of the Sprint. It is a 15-minute time-boxed event for the ST to synchronize activities with the SM and create a plan for the next 24 hours. This optimizes team collaboration and performance by inspecting the work since the last meeting and forecasting upcoming Sprint work. During this activity, these scenarios must be translated into BDD acceptance tests to guide the implementation, and ES can last several days until a given PI is obtained.

Table 8 summarizes the inputs, outputs, resources, roles and the main tasks of this activity.

Inputs	Selected BDD User Stories and Scenarios (Selected BDDUSS)
Outputs	Product Increment (PI)
Resources	Programming Languages (PL) Programming Tools (PTo) BDD BDD Tools (BDDT)
Roles	Software Engineer (SE) Scrum Master (SM) Scrum Team (ST)
Tasks	(a) Obtain BDD acceptance tests for the Selected BDDUSS; and (b) Perform the tasks to implement the Selected BDDUSS thus producing a Product Increment (PI).

Table 8: Inputs, outputs, resources, roles and tasks of ES activity

3.6.1 Acceptance Tests for the Selected BDDUSS of EAMS-CBALM

JBehave allows acceptance tests to be structure according to its textual story file. This file was created in the previous DSB activity, and contains the main behaviours of the feature set to be tested. Additional scenarios may be inserted if the SE believes it is necessary to complement the acceptance tests suit. There are six sets of acceptance tests in our user story example, one set for each instrument type to be registered, and the most relevant excerpts of the JBehave acceptance test code for the PATLP instrument type registration are shown in Figure 30.

JBehave uses the **@Given**, **@When** and **@Then** annotations to relate scenario specification clauses to Java methods, and the Java class that implements these methods should extend the Steps class. JBehave allows the scenario to be executed as a JUnit test (JUnit, 2016). The link between JBehave's executor framework and the textual scenarios is provided by the Embeddable class definitions. This class extends class JUnitStory and its name can be mapped to the textual story filename.

```

public class InstrumentRegisterSteps extends Steps{
... // setting variables and methods to support the scenarios
    @Given("the name is Performance Assessment of the Teaching-Learning Process ")
    public void setInstrumentName(String instrumentName){
        ... //set and check instrument name
    }
    @Given("the acronym is PATLP ")
    public void setInstrumentAcronym(String acronym){
        ... //set and check acronym role
    }
    @Given("the respondent is student")
    public void setRespondent(String respondent){
        ... //set and check respondent role
    }
    @Given("the appraisee is student")
    public void setAppraisee(String appraisee){
        ... //set and check appraisee role
    }
    @Given("the appraiser is teacher")
    public void setAppraiser(String appraiser){
        ... //set and check appraiser role
    }
    @When("clickOn Register")
    public void addInstrument(Button InstrumentRegisterAction) {
        ... //perform action
    }
    @Then("willBeCreated PATLP_Instrument")
    public void saveInstrument() {
        ... //data persistence
    }
    @Then("willBeCreated PATLP_FormHeader")
    public void createInstrumentFormatPHeader() {
        ... //creating, persisting format header for the instrument
    }
    @Then("loads the people performance evaluation format header page")
    public void updateInstrumentList() {
        ... //display instrument list updated
    }
}
// next scenario
...

```

Figure 30: Excerpt of the acceptance tests generated for the PATLP scenario

3.6.2 Implementation of the Selected BDDUSS of EAMS-CBALM

This task starts only after all acceptance tests related to the selected BDDUSS have been passed. Then, a readable behaviour-oriented code can be obtained since the ubiquitous language terminology of the application has been used in all ScrumOntoBDD activities. This can be verified in Figure 31, which shows excerpts of the JBehave implementation code for the Instrument Types Registration requirement. In this code, the names of the class and methods employ the EAMS-CBALM ubiquitous language terminology.

The complete JBehave implementation code represents the PI corresponding to the Instrument Types Registration requirement. Nevertheless, it must be inspected by the DT before being able to be delivered to the client.

```

1 package br.com.ufscar.dc.ges.pemaap
2
3 class Instrument {
4
5     String name
6     String acronym
7     Boolean dontHaveAppraisee
8     Boolean selfAppraisal
9     Boolean sameAppraiserAndRespondent
10    Boolean sameAppraiseeAndRespondent
11
12    static hasMany = [formats: Format]
13
14    static constraints = {
15        name blank: false, nullable: false
16        acronym blank: false, nullable: false
17        dontHaveAppraisee blank: false, nullable: false
18        selfAppraisal blank: false, nullable: false
19        sameAppraiserAndRespondent blank: false, nullable: false
20        sameAppraiseeAndRespondent blank: false, nullable: false
21
22        formats blank: true, nullable: true
23    }
24
25    static mapping = {
26        sort name: 'name'
27    }
28 }

```

```

8 class InstrumentService {
9
10    def messageSource
11
12    def create(data) {
13        Instrument instrument = new Instrument(
14            name: data.name,
15            acronym: data.acronym,
16            dontHaveAppraisee: data.dontHaveAppraisee,
17            selfAppraisal: data.selfAppraisal,
18            sameAppraiserAndRespondent: data.sameAppraiserAndRespondent,
19            sameAppraiseeAndRespondent: data.sameAppraiseeAndRespondent)
20        return instrument
21    }
22
23    def update(data) {
24        Instrument instrument = Instrument.findById(data.id as Long)
25
26        if (instrument) {
27            instrument.name = data.name
28            instrument.acronym = data.acronym
29            instrument.dontHaveAppraisee = data.dontHaveAppraisee
30            instrument.selfAppraisal = data.selfAppraisal
31            instrument.sameAppraiserAndRespondent = data.sameAppraiserAndRespondent
32            instrument.sameAppraiseeAndRespondent = data.sameAppraiseeAndRespondent
33        }
34
35        return instrument
36    }

```

Figure 31: Excerpt of the achieved implementation for the Instrument Types Registration requirement

3.7 Sprint Review and Retrospective (SRR)

The SRR activity starts with a DT meeting for inspecting the PI produced on the current sprint, and for adapting the PB if needed. This meeting is followed by another one for the DT to inspect itself, and for creating an improvement plan to be applied on the next sprint based on the SRA gathered during the current sprint. Table 9 summarizes the inputs, outputs, resources, roles and the main tasks of this activity.

Inputs	Product Increment (PI)
Outputs	Shippable Product (SP)
Resources	Sprint Records and Annotations (SRA)
Roles	Software Engineer (SE) Scrum Master (SM) Scrum Team (ST) Ontology Engineer (OE) Domain Expert (DE) Product Owner (PO)
Tasks	(a) Inspect the produced PI and adapt the PB if needed; and (b) Inspect the DT and create an improvement plan for the next sprint.

Table 9: Inputs, outputs, resources, roles and tasks of SRR activity

3.7.1 Sprint Review and Retrospective for EAMS-CBALM

An example of SRR activity for EAMS-CBALM that led to a PB adaptation, was the one related to the Instrument Types Registration requirement. During the DT meeting of this SRR activity, the PO reported:

“In addition to the student evaluation, the UFSCar Medicine Programme also contemplates the teacher evaluation, mainly when the latter is playing the facilitator or preceptor role in a curricular activity. In this case, similar instruments to those employed to evaluate students are also employed to evaluate teachers. For example, the PATLP instrument is also employed for teacher evaluation, but with different relationships between the involved actors: the respondent and appraiser is the student, and the appraisee is the teacher.”

Based on this report, the PATLP instrument specification, on the Evaluation Process ontology illustrated in Figure 21, was extended for also contemplating the teacher evaluation. Then, the PB item related to the Instrument Types Registration requirement was adapted for encompassing both, student and teacher evaluations. Figure 32 shows an excerpt of the Evaluation Process ontology with the extended PATLP instrument type allowing for student, facilitator and preceptor evaluations.

The screenshot displays the Protege ontology editor interface. On the left, the 'Class hierarchy: PATLP' pane shows a tree structure starting from 'owl:Thing'. Under 'EvaluationInstrument', several subclasses are listed: PATLP (highlighted), OSEPP, PBE, PT, RP, and WE. Other classes include ConstructivistSpiralSteps, CurricularActivity, EducationalActivity, EvaluationFormat, EvaluationType, FormatBody, FormatHeader, InteractionElement, LearningTrigger, Meeting, and Role.

The main area shows the 'Annotations: PATLP' pane with the following text:

Performance Assessment of the Teaching-Learning Process (PATLP):

The application of this evaluation is the responsibility of the facilitator and/or preceptor and should consider the established frameworks for a curricular

Below this, the 'Description: PATLP' pane shows the 'Equivalent To' section with a logical expression:

Equivalent To:

EvaluationInstrument

and ((EvaluationInstrument

and (theAppraiseeIs only Student)

and (theAppraiserIs only Facilitator)

and (theRespondentIs only Facilitator)) or

(EvaluationInstrument

and (theAppraiseeIs only Student)

and (theAppraiserIs only Preceptor)

and (theRespondentIs only Preceptor)) or

(EvaluationInstrument

and (theAppraiseeIs only (Facilitator or Preceptor))

and (theAppraiserIs only Student)

and (theRespondentIs only Student)))

Red boxes highlight the logical structure of this expression.

Figure 32: Evaluation Process ontology with extended PATLP

3.8 Final Considerations

In this chapter we presented the ScrumOntoBDD approach for agile software development, an approach that combines Scrum, Ontology and BDD. Throughout the chapter, the main activities of the proposed approach were presented and detailed in each section. To give a better understanding of ScrumOntoBDD, each activity was illustrated with excerpts of the case studies related to the EAMS-CBLAM development.

The case studies also gave evidences to verify hypotheses H_1 (Combining BDD with Scrum can improve this communication) and H_2 (Employing ontologies can eliminate these ambiguities). The case studies were developed by employing a ubiquitous language for the Education domain through BDD scenarios and acceptance tests. That allowed the PO to follow and properly communicate with the developers throughout the development process. When BDD artifacts (e.g. user stories and scenarios templates) were used inside the Scrum activities, the interaction between PO and developers increased and intensified, therefore, improving their communication (Souza, 2017). Also domain ontologies were used to reduce the ambiguity introduced by using natural languages to report user stories. Starting from a reference ontology, the UFSCar Medicine Programme ontology was defined. From there, the Evaluation Process ontology was specialized and used with the user stories for determining the features to be developed in the EAMS-CBALM evaluation module. By using ontologies during all development, it was possible to define a common language used by all actors involved and minimized ambiguities when defining and implementing the evaluation module feature sets (Souza, 2018).

The following chapter will describe an experimental analysis of ScrumOntoBDD to evaluate these hypotheses when employing our approach. After an introduction to Action Research, the process used for this analysis, the experiment is described according to the Action Research cycle and phases, and its results will be discussed in regard of our hypotheses.

Chapter 4

SCRUMONTOBDD ANALYSIS

This chapter presents an experimental analysis of ScrumOntoBDD, whose main objective is to evaluate our two hypotheses when employing this approach. Section 4.1 introduces Action Research, the process used in this analysis. Section 4.2 presents the objective, settings, and participants of the experiment for doing this analysis. Section 4.3 describes the experimental analysis focusing on its AR phases. Section 4.4 has the final considerations

4.1 Action Research

Research can be theoretical or practical and, according to Nuzhat (Nuzhat, 2011), two major categories of practical research are empirical research and developmental or problem-solving research. The first one is evidence-oriented focusing on data collection, and the researcher seeks mainly to describe or theorize rather than develop or prescribe. The second one is utility or solution oriented, and the researcher participates in the involved activities, typically looking for developing or prescribing artefacts that will be an integral part of the acquired knowledge with that research. Design Research (DR) and Action Research (AR) belong to this second category.

In DR, the design process may be the research object, or an instrument of that research. In the latter case, the research activities can include the designing, possibly

the building, and the design evaluation of artefacts, to understand or improve design. These artefacts or their design are outputs of this kind of research.

In AR, the focus is mainly on practice improvement not only on knowledge increase, and the involved actors in the practice being investigated are engaged in the research activities. This kind of research includes data collection, but in the context of plans and actions for problem-solving. According to Järvinen (Järvinen, 2007), considering the utility aspect, and that researcher and practice actors work together in the research activities, AR is similar to DR.

As illustrated in Figure 33, AR is a cyclical process composed of five phases: *Diagnosing*, for identifying or defining a problem; *Action Planning*, for considering alternative courses of action to solve a problem; *Action Taking*, for selecting a course of action; *Evaluating*, for studying consequences of an action; and *Specifying Learning*, for identifying general findings.

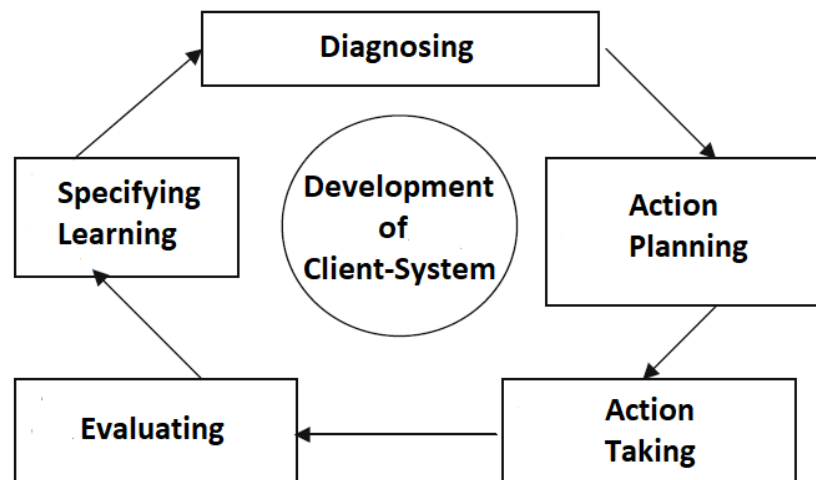


Figure 33: Action Research cycle (adapted from (Järvinen, 2007))

This AR cycle can be performed several times for achieving a problem solution, and AR projects may differ in the number of phases that is carried out for solving a problem. In Järvinen (Järvinen, 2007), this cycle is considered to be a canonical one, and from its phases he concludes: AR means both action taking and evaluating; and AR is carried out in collaboration between researcher and system client.

We use AR to carry out the experimental analysis of ScrumOntoBDD, since this cyclical process is guided by the collaborative and continuous learning principles, and due to the cooperative and participative involvement between the researcher and the actors of this experiment.

4.2 Objective, Settings, and Participants

The main objective of this experiment is to evaluate if ScrumOntoBDD allows for both, improving quite a lot the communication between PO and developers (H_1), and eliminating quite a lot the ambiguities intrinsic of using natural languages to report user stories (H_2).

We have conducted this experiment outside professional and academic environments with the participation of the same PO and the same PhD's student, who were involved in the EAMS-CBALM development by the software house, for being able to evaluate our hypotheses. Together, we choose a usage scenario in the Education domain, inside the context of the UFSCar Medicine Programme, and supported by EAM-CBALM, which has similarities to the one employed to illustrate ScrumOntoBDD. Thus, we could reuse most of the ontologies already created and validated, but this new usage scenario should be more complex, and more often requested by system's users than the first one.

So, we chose a usage scenario for this experiment also supported by the Evaluation Management module of EAMS-CBALM, but related to a more comprehensive functional requirement of this module: *Create People Performance Evaluation Format* requirement.

The participants of this experiment were the following: the PO involved in the EAMS-CBALM development, who also played the DE role since she was a teacher of the UFSCar Medicine Programme; the PhD's student involved in the EAMS-CBALM development, who played the SE role; and the author of this MSc's dissertation, who was the AR researcher, and played all the other roles defined in ScrumOntoBDD. Besides these participants, an AR expert participated in the preparation and analysis

of a semi-structure interview, which was employed in the Evaluating and Specifying Learning phases of the AR cycles. All these participants signed *The Written Informed Consent Form*, whose original model in Portuguese is in Appendix A, and their profiles are summarized below:

(a) *Participant 1 (PO)* holds a Medical's degree from Faculty of Medicine of Marília (Famema) in 1984, a MSc's degree in Public Health - Epidemiology from Faculty of Public Health (FSP) of University of São Paulo (USP) in 1997, and a Professional MSc's degree in Evidence-Based Medicine Methodology from Paulista Medical School (EPM) of Federal University of São Paulo (UNIFESP) in 2002. She was Assistant Professor of DMed/UFSCar until March 2017 where she retired. She has experience in the areas of Collective Health and Health Professional Education, working in information systems, evaluation of teaching and health services, academic management of undergraduate programmes and Lato Sensu graduate programmes, teacher training in active learning methodologies, and continuing education of health professionals;

(b) *Participant 2 (SE)* holds a Bachelor's degree in Data Processing Technology from State University of Maringá (UEM) in 1985, and a MSc's degree in Computer Science from the Graduate Programme on Computer Science (PPG-CC) of Federal University of São Carlos (UFSCar) in 2001. Currently, she is a PhD's student at PPG-CC/UFSCar, a Professor from Federal Institute of Education, Science and Technology of São Paulo (IFSP) at Birigui campus, and her main interests are in the areas of Software Engineering, Computer Supported Cooperative Work, Learning Management Systems, Ubiquitous Computing, and Virtual and Augmented Realities.

(c) *Participant 3 (AR researcher)* holds a Bachelor's degree in Biomedical Informatics from University of São Paulo (USP) in 2009, and a Specialization in Computer Science and Computational Mathematics from USP in 2010. Currently, he is a MSc's student at PPG-CC/UFSCar, and his main interests are in the areas of Software Engineering, Database, Health Informatics, and Education Informatics; and

(d) *Participant 4 (AR expert)* holds a Bachelor's degree in Letters from Federal University of Minas Gerais (UFMG) in 1975, a Master's degree and a PhD's degree in Linguistics, Literature, and Arts from University of Montpellier III (France) in 1977

and in 1980, respectively, a post-doctoral from University of Montreal in 1985 and a post-doctoral from University of Toronto in 2002, both on Linguistics, Literature and Arts. Currently, she is Full Professor of Department of Applied Linguistics at the Institute of Language Studies of State University of Campinas (UNICAMP), and her main interests are in the areas of Literacy, Teacher Training, Language and Identity, and Languages and Technologies.

4.3 Experimental Analysis

According to Lewin (Lewin, 1988), in AR the work is partitioned into stages and can involve several cycles, beginning with recognition, for example of a problem, following by data collection, analysis and development of hypotheses. This leads to a second cycle in which the hypotheses are evaluated in practice and the changes analyzed. This cyclical process does not come to a natural conclusion, although at some point it must be closed and its results should be disseminated.

Similar to Mejía-Gutiérrez (Mejía-Gutiérrez, 2017), we conducted our experimental analysis through two AR cycles: the first one employing Scrum, and the second one employing ScrumOntoBDD. The research questions **RQ₁** and **RQ₂** were the analysis input, and the evaluation of hypotheses **H₁** and **H₂** were the analysis output. Figure 34 illustrates these two AR cycles with their phases.

In order to have PO and SE independent views of this experiment, we executed it twice: first involving Participant 1 and Participant 3; and second involving Participant 2 and Participant 3. The AR phases in Figure 33 were performed as follows:

(a) *Diagnosing* is the problem definition to be investigated by this experiment, i.e., the research questions **RQ₁** and **RQ₂** for the first cycle;

(b) *Action Planning* starts by choosing the EAMS-CBALM usage scenario and the Application Domain Documents (ADD) for this experiment, which are the same for both cycles, followed by planning Scrum activities for the first cycle, and ScrumOntoBDD activities for the second cycle;

(c) *Action Taking*, for the first cycle is the execution of the Scrum activities for achieving an implementation of the EAMS-CBALM usage scenario, which was performed by the software house, and for the second cycle is the execution of the ScrumOntoBDD activities for achieving another implementation of the same EAMS-CBALM usage scenario, which was performed by the AR researcher;

(d) *Evaluating* is a semi-structured interview performed in both cycles and conducted by Participant 3, where the interviewee was the Participant 1 in the first execution of the experiment, and was the Participant 2 in the second one; and

(e) *Specifying Learning* is the analysis of the data collected in the interview, which in the first cycle will be a feed for the *Diagnosing* phase of second cycle, and in the latter will be the output of the whole experiment, i.e., evaluation of hypotheses H_1 and H_2 .

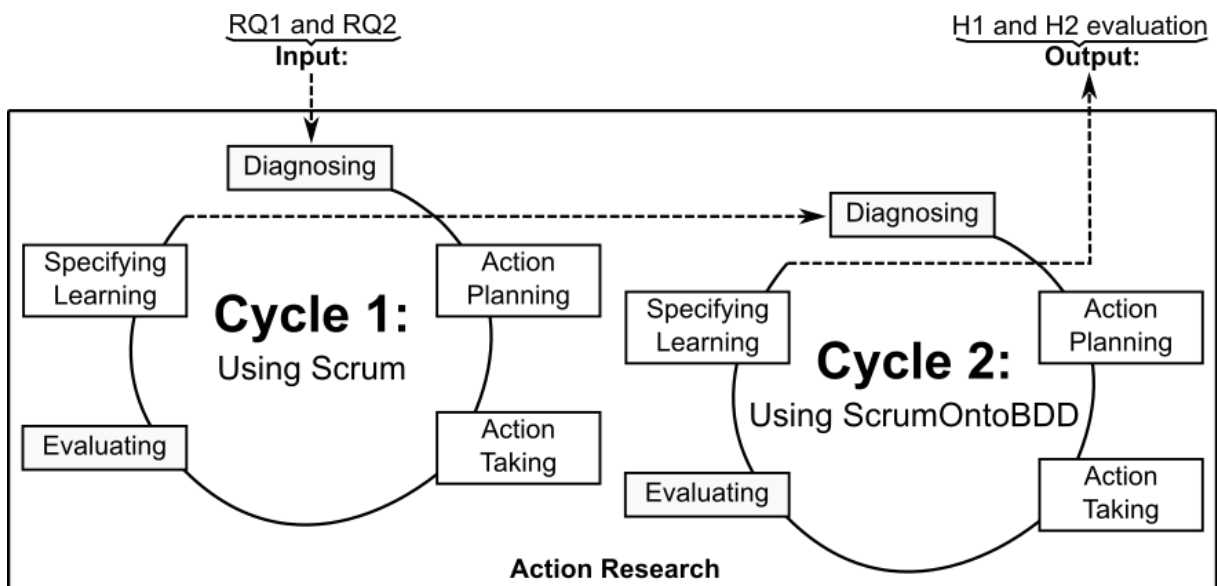


Figure 34: AR cycles of experimental analysis (adapted from (Mejía-Gutiérrez, 2017))

4.3.1 Action Planning

The EAMS-CBALM usage scenario chosen for this experiment corresponds to the *Create People Performance Evaluation Format* requirement, which was extracted from the following user story reported by the PO during the EAMS-CBALM development:

“In order for doing a PATLP evaluation, the PATLP evaluation format must be created by the programme coordinator. For accomplishing that, first it is necessary to create the category of this evaluation format, which in this case is People Performance Evaluation, then define the course, offer, class, curricular activity, educational activity, and education action.

For registering a PATLP evaluation, the format must keep the following fields: the format title; the instrument type that was previously registered; a descriptive field for reporting observations; the period with start and end dates that the format will be available to be answered; and a confirmation of which actors should carry out the evaluation.

After defining the format header, the coordinator defines the format body, which consists of questions of essay type or objective type of single answer. The format body must have a field called concept that, according to the relationships between the involved actors, may accept one of the following concepts: satisfactory, or needs improvement, or unsatisfactory, when the appraise is the student; satisfactory, or needs improvement, or without concept emission, when the appraise is the teacher.

The system should allow the reuse of previously registered format bodies that may be editable. In addition, the questions of these imported bodies should be in red for signaling they have been reused. When ending the edition of an imported body, the red color should be changed to the default one for signaling this body has been successfully validated. Examples of questions contained in a format body when the the appraise is the facilitator:

Essay type questions	<i>How was the facilitator performance in the provisory synthesis and in the new synthesis? Justify. How was the fulfillment of the work pacts? Justify. Additional comments and/or suggestions from the student to the facilitator.</i>		
Objective type questions of single answer	<i>How was the facilitator performance in the provisory synthesis and in the new synthesis? <input type="checkbox"/> Excellent <input type="checkbox"/> Good <input type="checkbox"/> Average <input type="checkbox"/> Bad</i>	<i>How was the fulfillment of the work pacts? <input type="checkbox"/> Excellent <input type="checkbox"/> Good <input type="checkbox"/> Average <input type="checkbox"/> Bad</i>	<i>Other specific questions of this activity. <input type="checkbox"/> Excellent <input type="checkbox"/> Good <input type="checkbox"/> Average <input type="checkbox"/> Bad</i>
Concept	<i><input type="checkbox"/> Satisfactory <input type="checkbox"/> Needs improvement <input type="checkbox"/> Without concept emission</i>		

“

Based on this user story and the ADD Medicine Programme - CCBS Pedagogical Political Project (UFSCar, 2007), the activities for both AR cycles were planned. Figure 35 gives an overview of this planning, and it should be highlighted

that the *Screen Prototypes (SPr)* activity was included in Scrum by the software house.

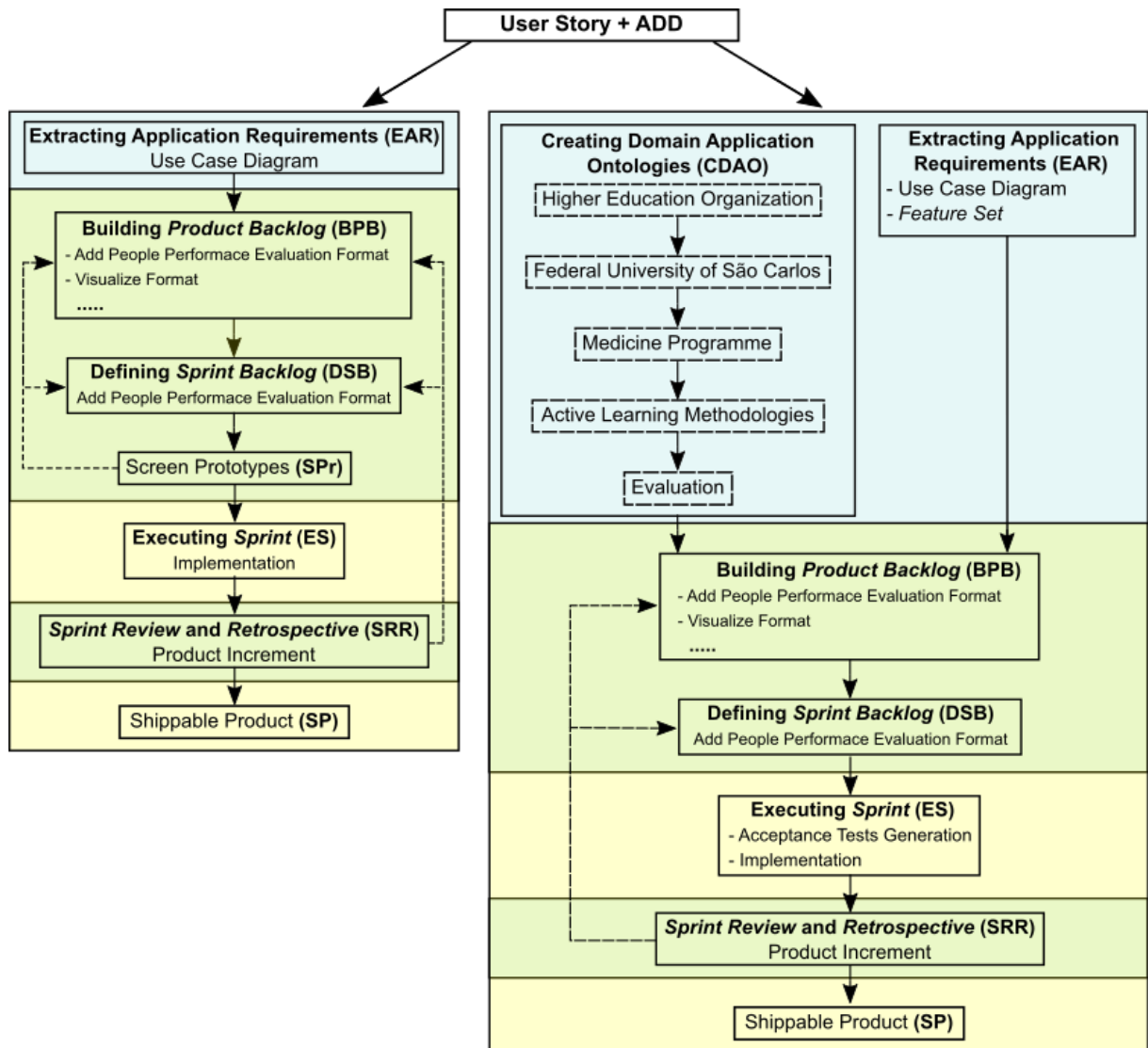


Figure 35: Activities in the Action Planning phases of the AR cycles

4.3.2 Action Taking

Except for the SPr activity in Cycle 1 and for the CDAO activity in Cycle 2, all remain activities are common for both cycles, and the difference between them is the way they were taken. So, in this section, it will be described and illustrated the main tasks involved in the execution of these activities highlighting their differences.

The first task of the EAR activity was the same for both cycles, i.e., identification of the application scenarios by means of a UML use case diagram. Figure 36 shows an excerpt of this diagram that corresponds to the Create People Performance Evaluation Format requirement, which was extracted from the user story reported by the PO.

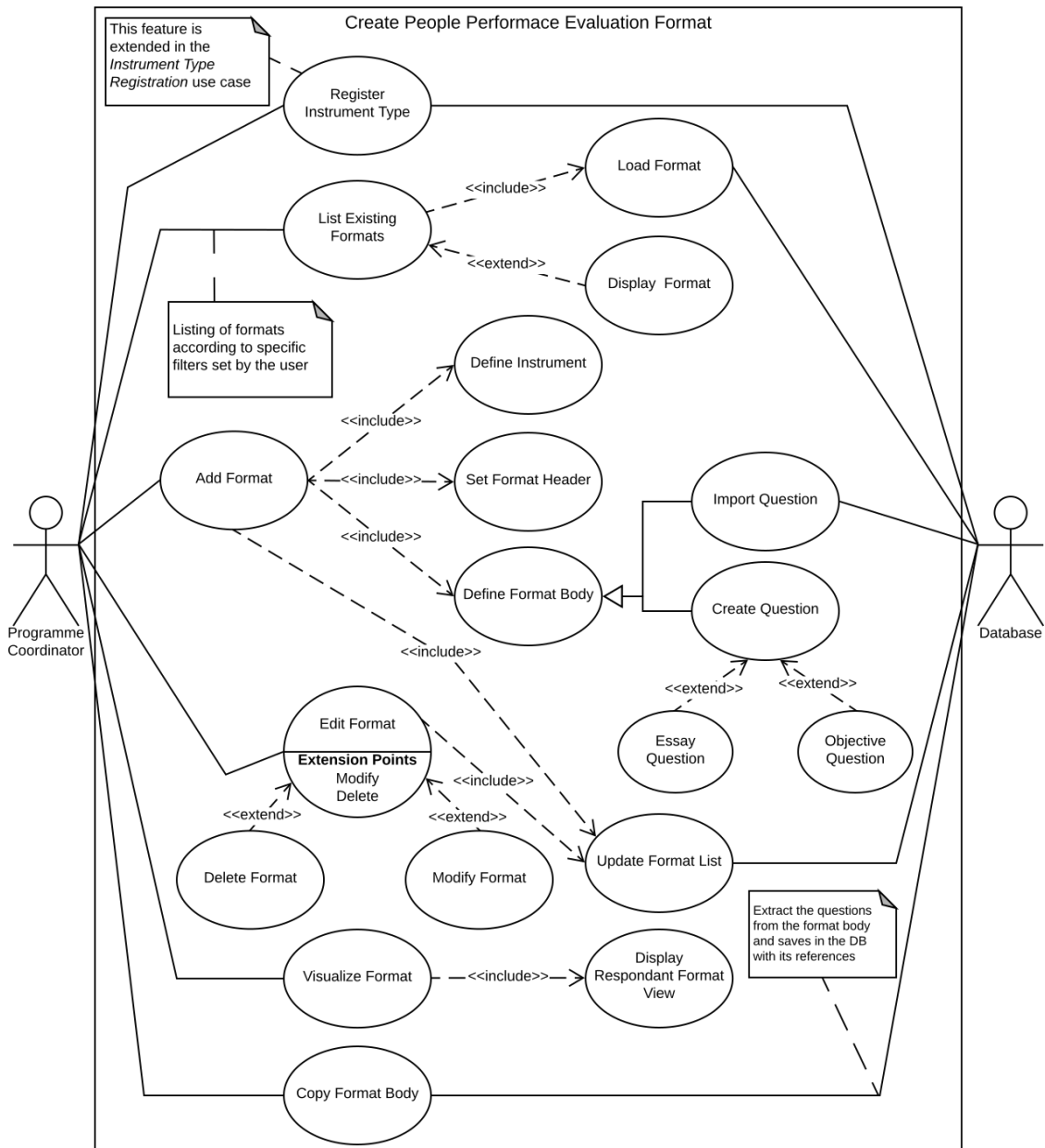


Figure 36: Excerpt of UML use case diagram for Create People Performance Evaluation Format requirement

Only the EAR activity of Cycle 2 have had a second task since it is BDD related: the feature set definition based on the application scenarios described in the

UML use case diagram. An excerpt of the *Add People Performance Evaluation Format* feature is illustrated in Figure 37.

Feature: *Add People Performance Evaluation Format*

In order *for the programme student to carry out a PATLP evaluation*

As a *programme coordinator*

I want to *add a new People Performance Evaluation Format into the system*

To facilitate the this feature comprehension, we refine it into three large sets of scenarios that represent the main user behaviors:

- Selection of the target group for the evaluation format that will be created.
- The settings of the format header.
- The settings of the format body.

Background 1:

Given the following programmes and its respective offers exist:

Programme	Offer
Medicine	Offer 2015 Offer 2016
Postgraduate Program in Clinical Management	Offer PPCM

Given the following class exist:

Offer	Class
Offer 2015	Class X (2015)
Offer 2016	Class XI (2016)
Offer PPCM	VII – 2017

Figure 37-A: Excerpt of the Add People Performance Evaluation Format feature

Given the following curricular activities and its respective educational actions exist:

Programme	Educational Unit	Curricular Activity	Educational Action
Medicine	Health Needs and Therapeutic Plans I	Simulation Stations I	- Workshops - Simulations
		Situations-Problem I	- Workshops (formulating explanations) - Problem-Situation Processing
	Professional Practice I	Family and Community Health I	- Practice at FHU - Practice Reflection
PPCM	---	Care Lines and Dot Matrix Support	- ABP - Workshops - Plenary - Portfolio

Scenario 1.1: *the programme coordinator selects target group for the new evaluation format that will be created*

Given *the programme coordinator selects People Performance Evaluation*

And *selects the Medicine programme*

And *selects the Offer 2016*

And *selects the Class XI (2016)*

And *selects the curricular activity Simulation Stations I*

When *click on the educational action Workshops*

Then *the system verify the selected data integrity*

And *loads the people performance evaluation format header page*

Scenario 1.2 : *scenario 1.1 variations*

...

Background 2 (for settings the format header):

...

Figure 37-B: Excerpt of the Add People Performance Evaluation Format feature

Almost all the ontologies created for illustrating ScrumOntoBDD have been reused, adapted or extended in the CDAO activity of Cycle 2 of this experiment. For instance, the Evaluation Process ontology, whose excerpts are shown in Figures 20 and 21, was extended for allowing the description of the Create People Performance Evaluation Format requirement. Figure 38 shows an excerpt of the Extended Evaluation Process ontology.

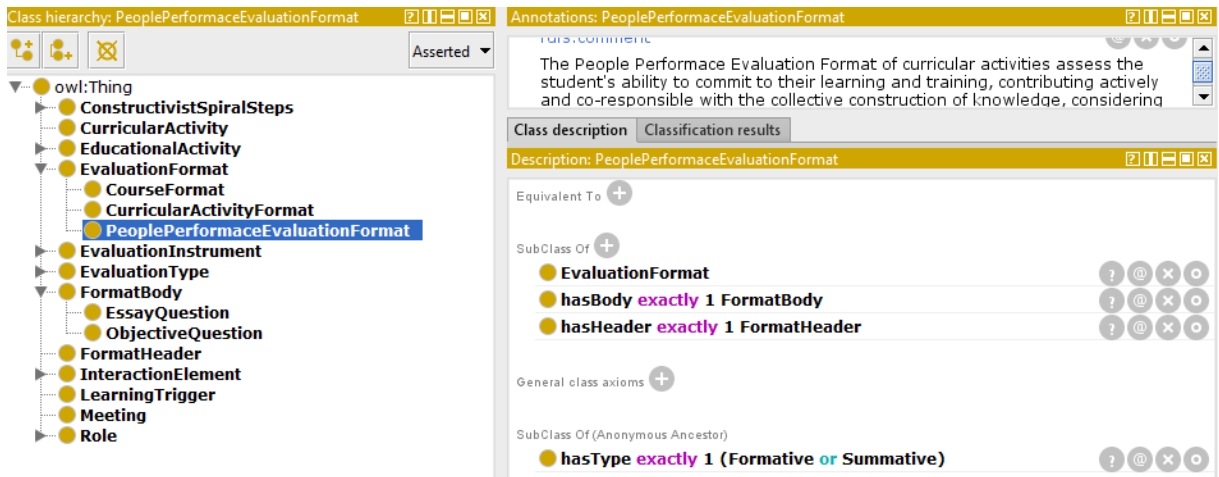


Figure 38: Excerpt of Extended Evaluation Process ontology focusing on People Performance Evaluation Format

The main difference between the BPB activities of Cycles 1 and 2 is that on the first the PB items were informally specified using tables, while on the second they were formally specified using ontologies. Figure 39 shows an excerpt of the PB item selected in the DSB activity of Cycle 1, which was specified by the software house during the EAMS-CBALM development, while Figure 40 shows an excerpt of the PB item selected in the DSB activity of Cycle 2, which was specified by the AR researcher of this experiment.

Project	Key	Summary	Issue Type	Status	Priority	Resolution	Assignee	Sprint
PEMAAP-Sirio-Libanés	SIR-2900	SIR-2874 [Front] Implement format body copy	Sub-task	Resolved	Major	Done	Developer 1	Sprint 16, Sprint 17
PEMAAP-Sirio-Libanés	SIR-2899	SIR-2874 [Front] Implement group membership to the People Performance Evaluation format	Sub-task	Closed	Major	Done	Developer 1	Sprint 16, Sprint 17
PEMAAP-Sirio-Libanés	SIR-2785	SIR-2080 [Front] Format creation / editing to allow questions editing	Sub-task	Closed	Major	Done	Developer 2	Sprint 16, Sprint 17
PEMAAP-Sirio-Libanés	SIR-2661	SIR-1849 [Front] In the format, in question registration, create 2 types of questions: essay and objective	Sub-task	Closed	Major	Done	Developer 2	Sprint 16, Sprint 17
PEMAAP-Sirio-Libanés	SIR-2636	SIR-2601 [Front] After saving a format, return to the format query screen	Sub-task	Closed	Major	Done	Developer 1	Sprint 16
PEMAAP-Sirio-Libanés	SIR-2601	As PO, I wish all bugs related to Formats to solved	Bug	Closed	Major	Done	Unassigned	Sprint 17

Figure 39: Excerpt of the PB item selected in Cycle 1

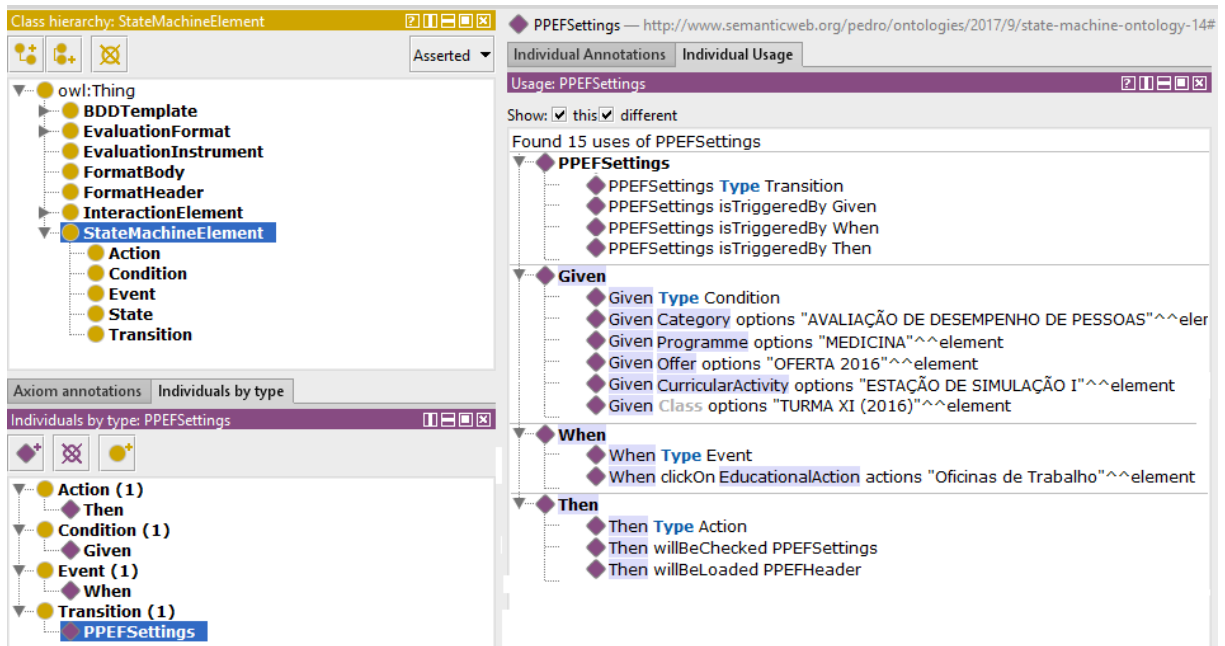


Figure 40: Excerpt of the PB item selected in Cycle 2 focusing on the selection of the target group for the evaluation format scenario

SPr activity was included in Cycle 1 because it was widely used by the software house throughout the EAMS-CBALM development, and was not included in Cycle 2 because we wanted to assess its importance to eventually include it in ScrumOntoBDD. Figure 41 shows one of the screen prototypes related to the selected PB item and employed in Cycle 1 of this experiment.

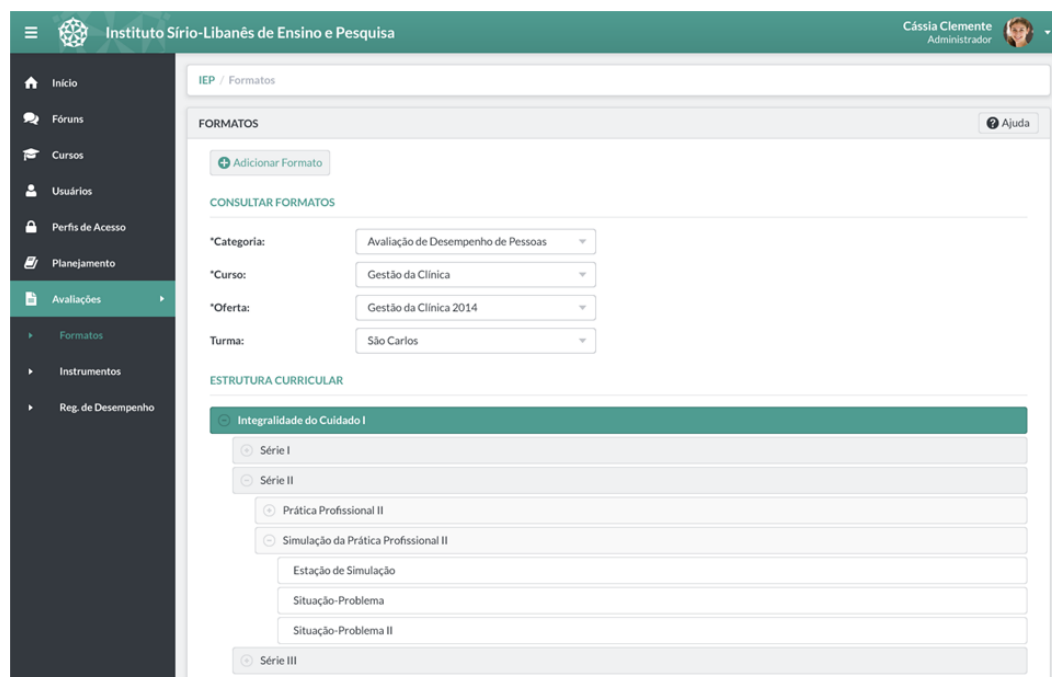


Figure 41: Screen prototype for the selected PB item

The ultimate goal of ES activity in both cycles is to obtain an implementation of the selected PB item. However, in Cycle 2 the Acceptance Tests Generation, a BDD related task, was performed before the Implementation task itself. Figure 42 shows an excerpt of the acceptance tests generated in Cycle 2 for the selected PB item. Only the methods names were kept to facilitate BDD Acceptance test characteristics.

```

public class PPEFSettingsSteps extends Steps{

... // setting variables and methods to support the scenarios

    @Given("the programme coordinator selects People Performance Evaluation")
    public void setCategory(DropdownButton Category){
        ... //set Evaluation format Type
    }

    @Given("selects the Medicine programme")
    public void setProgramme(DropdownButton Programme){
        ... //set Programme for this format
    }

    @Given("selects the Offer 2016")
    public void setOffer(DropdownButton Offer){
        ... //set relationship
    }

    @Given("selects the Class XI (2016)")
    public void setClass(DropdownButton Class){
        ... //set relationship
    }

    @Given("selects the curricular activity Simulation Stations I")
    public void setCurricularActivity(Grid CurricularActivity){
        ... //set relationship
    }

    @When("click on the educational action Workshops")
    public void setEducationalAction(Button EducationalAction) {
        ... //perform action
    }

    @Then("the system verify the selected data integrity")
    public void checkDataIntegrity() {
        ... //Check inserted data for black spaces or no Classes
    }

    @Then("loads the people performance evaluation format header page")
    public void displayPPEFHeader() {
        ... //creating, persisting and display format header
    }

// next scenario
...
}

```

Figure 42: Excerpt of the acceptance tests generated in Cycle 2

Although the implementations for both cycles correspond to the same system requirement, the actions taken to get them followed different methodologies. Figure 43 shows an excerpt of the achieved implementation on Cycle 1, which was obtained by the software house during the EAMS-CBALM development, while Figure 44 shows an excerpt of the achieved implementation on Cycle 2, which was obtained by the AR researcher of this experiment. These implementations correspond to the PIs to be analysed in the SRR activities of these cycles. Although both implementations have similar structural code, they may have different names for classes, attributes and methods. Since Cycle 2 uses ScrumOntoBDD approach, its code terminology is closer to the application domain. Therefore, this implementation should be easier for understanding by the participants.

```

1 package br.com.tokenlab.pemaap
2
3 class Format {
4
5     String topic
6     Long bracket
7     String historical
8     Long assessmentBracket
9     Boolean mustBePublished
10    String notes
11    Long numberOfImprovementPlans
12    Date timeLengthInitialDate
13    Date timeLengthFinalDate
14    Date answerTimeInitialDate
15    Date answerTimeFinalDate
16    Date commentTimeInitialDate
17    Date commentTimeFinalDate
18    Date validationTimeInitialDate
19    Date validationTimeFinalDate
20
21    static hasOne = [appraisee: OfferOccupation,
22                    evaluator: OfferOccupation,
23                    course: Course,
24                    instrument: Instrument,
25                    levelFour: LevelFour,
26                    levelFive: LevelFive,
27                    intervieweeWithOfferOccupation: OfferOccupation]
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Figure 43: Excerpt of the achieved implementation on Cycle 1

Chapter 4: ScrumOntoBDD Analysis

```
1 package br.com.ufscar.dc.ges.pemaap
2
3 class Format {
4
5     String title
6     String historicalNotes
7     Long category
8     Long assessmentCategory
9     Long numberOfImprovementPlans
10    Date relatedPeriodInitialDate
11    Date relatedPeriodFinalDate
12    Date answerPeriodInitialDate
13    Date answerPeriodFinalDate
14    Date commentPeriodInitialDate
15    Date commentPeriodFinalDate
16
17    SortedSet questionsStatements
18
19    static hasOne = [appraisee: OfferOccupation,
20                    appraiser: OfferOccupation,
21                    course: Course,
22                    instrument: Instrument,
23                    levelFour: LevelFour,
24                    levelFive: LevelFive,
25                    respondentWithOfferOccupation: OfferOccupation]
26
27
28
29
30
31
32
33
34
35
36
37    static mapping = {
38        historicalNotes type: 'text'
39    }
40
41    static constraints = {
42        title blank: false, nullable: false
43        historicalNotes blank: true, nullable: true
44        category blank: false, nullable: false
45        assessmentCategory blank: false, nullable: false
46        numberOfImprovementPlans blank: false, nullable: false
47
48        relatedPeriodInitialDate blank: true, nullable: true
49        relatedPeriodFinalDate blank: true, nullable: true
50        answerPeriodInitialDate blank: false, nullable: false
51        answerPeriodFinalDate blank: false, nullable: false
52        commentPeriodInitialDate blank: false, nullable: false
53        commentPeriodFinalDate blank: false, nullable: false
54
55        appraisee blank: true, nullable: true
56        appraiser blank: true, nullable: true
57        course blank: false, nullable: false
58        instrument blank: false, nullable: false
59        levelFour blank: true, nullable: true
60        levelFive blank: true, nullable: true
61        respondentWithOfferOccupation blank: true, nullable: true
62    }
63
64    def retrieveFormatsForActivityAppraisal(formatQueryResultsTreeMap) {
65        JSONArray formatsForActivityAppraisalByLevelAndCourseAndOfferAndSchoolClassJSONArray = new JSONArray()
66
67        formatQueryResultsTreeMap.values().each { formatQueryResult ->
68            JSONArray formatsJSONArray = new JSONArray()
69            JSONObject formatsJSONObject = new JSONObject()
70            formatsJSONObject.put('listTitle', FORMATS + formatQueryResult.title)
71            formatsJSONObject.put('formats', formatsJSONArray)
72            formatsForActivityAppraisalByLevelAndCourseAndOfferAndSchoolClassJSONArray.add(formatsJSONObject)
73
74            formatQueryResult.formats.each { format ->
75                JSONObject formatJSONObject = new JSONObject()
76                formatJSONObject.put('formatId', format.id)
77                formatJSONObject.put('formatTitle', format.title)
78                formatJSONObject.put('formatPeriodInitialDate', format.periodInitialDate)
79                formatJSONObject.put('formatPeriodFinalDate', format.periodFinalDate)
80                formatJSONObject.put('formatAnswerPeriodInitialDate', format.answerPeriodInitialDate)
81                formatJSONObject.put('formatAnswerPeriodFinalDate', format.answerPeriodFinalDate)
82                formatsJSONArray.add(formatJSONObject)
83            }
84        }
85    }
86 }
```

Figure 44: Excerpt of the achieved implementation on Cycle 2

4.3.3 Evaluating

The semi-structured interview model was chosen since it follows a pre-defined script of subjects to be addressed with the interviewee, instead of a sequence of issues of questionnaire type. This allows the interviewee to more freely report his or her experiences and more freely articulate his or her thoughts about them.

The interview with Participant 1 (PO) was held on 06 August 2018 in two sections with a total time of 01h 45' 50", and with Participant 2 (SE) the day after in a single section of 01h 45' 14". These interviews were audio-taped with the participants consent, their original script in Portuguese is in Appendix B, and their excerpts transcribed also in Portuguese are in Appendix C.

Each interview was composed of two parts corresponding to the AR cycles of the experiment. In the first part (Cycle 1), the AR researcher presented and

illustrated, by means of the implementation development of the Performance Evaluation Format requirement of EAMS-CBALM, the Scrum activities employed by the software house for developing this system, which resulted in the product currently being tested in the UFSCar Medicine Programme. These explanations were always gone along with comments by the interviewee regarding both, his or her participation in the EAMS-CBALM development, and the Scrum method employed in this development. Furthermore, the AR researcher was always encouraging the interviewee to provide additional remarks, clarifications, and criticisms about these matters.

In the second part (Cycle 2), the AR researcher presented and illustrated, by means of the implementation development of the same EAMS-CBALM requirement, the ScrumOntoBDD activities. The same dynamics of interview part 1 occurred here, i.e., the interviewee was always spontaneously providing comments, and the AR researcher was always encouraging she to refine these comments, and to carry out with ScrumOntoBDD evaluations, comparing them with those of Scrum done in Cycle 1, and also taking into account his or her professional experience.

These audio-taped interviews were fully transcribed in Portuguese using conventional writing, the transcriptions were analysed, their most relevant excerpts were selected, and these excerpts were grouped according to their Cycles and according to the main results of this analysis. The English translation of each one of these excerpts is presented bellow followed by its original transcription, and employing the following conventions: word fragments, hesitations, repetitions, overlapped speeches and words or phrases which cannot be reliably identified are flagged by (...); and additions of meaning explicitness are flagged by [].

Transcription excerpts related do Cycle 1:

Agility due to weekly meetings (PO)

00h 14' 30" - 00h 14' 39"

"The methodology is quick, you can make things move faster due to weekly meetings, but there is another side: you can lose things. (...)"

"A metodologia é rápida, você consegue fazer as coisas caminharem mais rapidamente ao se reunir semanalmente, mais tem esse outro lado: que você pode perder mesmo coisas. (...)"

Lack of more systematic "records" that could give more visibility to the different development phases (PO)

00h 12' 30" - 00h 13' 04"

"So, the system rules, they were not transparent to me. So, there are a lot of rules that, from what I can see here [Product Backlog] (...) I did not really know what rules they were going to implement and they had things implemented that were not what I'd wanted, you know what I mean?"

"Então, são regras do sistema, que elas não ficaram transparentes pra mim. Então tem várias regras, pelo que eu tô vendo aqui [Product Backlog] (...) que na verdade eu não fiquei com a certeza de que regras que eles iam implementar e teve coisas que eles implementaram que não eram o que eu gostaria de ter, entendeu?"

00h 13' 31" - 00h 13' 54"

"things that may not have been clear and when they pass it over here [pointing to EAR] and when they do it [pointing to BPB], if they have any doubts and I did not explain, they make a decision and follow through and afterwards I will just find out that wasn't really it (...) Or it is not clear which rule is valid at the moment. So now that I'm testing it, I'm working on checking what the system is really doing but what was said o the developers to do is not clear to me."

"coisas que possam não ter ficado claras e na hora que eles passam pra cá [aponta para o levantamento de requisitos] e na hora que eles vão fazer [aponta para o Product Backlog], se eu não falei e eles têm dúvida, eles tomam uma decisão e fazem e depois que eu vou descobrir que não era bem aquilo (...) Ou fica sem transparência de qual é a regra que tá valendo. Então agora que eu estou testando, estou fazendo um trabalho de checar o que o sistema tá fazendo mas sem ter clareza daquilo que foi passado para eles fazerem mesmo."

00h 14' 00" - 00h 14' 13"

“So, I think that in this passage here [BPB], I think it's a moment in which you can lose, you can lose things. Or things that are going to be misleading, and then they'll need to be fixed later, right? A need for correction.”

“Então, eu acho que nessa passagem aqui [criação do product backlog], eu acho que é um momento em que se perde, podem se perder coisas. Ou coisas que vão ficar com equívoco e depois vão precisar de acerto depois, né? De correção.”

Using agile software development methodologies like Scrum (SE); and

Scrum distributes roles and activities to team members, but does not explain how to implement these activities (SE)

00h 04' 32" - 00h 04' 57"

“The use of an agile methodology helped a lot to develop the system (...) Scrum is great for organizing functions within the team, however, after you have distributed the roles, you have to tell how to do the activities, and this Scrum does not do”

“A utilização de uma metodologia ágil facilitou muito o desenvolvimento do sistema (...) o Scrum é ótimo pra você organizar funções dentro da equipe mas depois que você dividiu papéis, você tem que falar como fazer as atividades e isso o Scrum não faz”

Lack of a formal methodology that facilitates the communication with the user and with the developer (SE)

00h 06' 07" - 00h 06' 29"

“And I think a lot of things stay in R.'s head [the guy responsible for the development team] (...) I did not see how he externalizes what he managed to grasp from what the PO gave to him (...) so much that they did not make it available, you saw that, right? from that pile of material that they have (...) only the ER model [Entity-Relationship model] was available.”

“E eu acho que muita coisa fica na cabeça do R. [responsável pela equipe de desenvolvimento] (...) eu não vi como que ele exterioriza aquilo que ele conseguiu captar ali do que a PO passou pra ele (...) tanto que eles não disponibilizaram, você viu, né, naquele monte de material que tem (...) só o MER [Modelo Entidade-Relacionamento].”

00h 07' 56" - 00h 09' 32"

"So, he [software house developer] used to do it in the powerpoint, he used to do the screens browsing and come back to talk to her [PO], but to validate what she had said at the first moment (...) then she would talk, talk and talk, and he would understand, process, and write it in some type of wordpad editor (...) after that he would build up the ER model and those browsing prototypes, and at the next meeting he would bring them to discuss with her to validate if what she had spoken, he had understood.

"Então, ele [engenheiro de software] fazia no powerpoint, ele fazia a navegação da tela e voltava pra conversar com ela, mas pra validar aquilo que ela [PO] tinha falado no primeiro momento (...) então, ela contava, contava, contava, ele entendia, ele processava, ele escrevia num editor de texto tipo word pad (...) depois disto ele montava o MER esses protótipos navegáveis, e no próximo encontro trazia pra discutir com ela, pra validar se o que ela tinha falado, ele tinha entendido."

00h 41' 45" - 00h 42' 17"

"Even though she understands all this process, she always tried to detail everything, but sometimes a little something did not show up and then, that little something would only show up when she saw the prototype (...) and then, she would realize that we did not understand, so she would say: let's open the IntegraMed, then, we would open the IntegraMed and discuss it with them."

"Mesmo ela entendendo todo esse processo, ela sempre tentou detalhar tudo, mas às vezes alguma coisinha num vinha e aí essa coisinha só vinha quando via o protótipo (...) aí que ela via que a gente não estava entendendo, ela falava: vamos abrir o IntegraMed, aí abria o IntegraMed e discutia com eles."

Division into non-testable sub-tasks throughout the process

00h 20' 12" - 00h 20' 35"

"What kind of difficulty did we have at the time? The whole process is too large. So we have to break It (...) It was broken (...) maybe the lack of experience from the developers, where I put myself, in this new way of evaluating and testing each part"

"que que a gente teve de dificuldade na época? É que o processo todo é muito grande. Então a gente tem que quebrar (...) Foi quebrado (...) talvez a falta de experiência dos desenvolvedores, onde eu me coloco, nessa nova forma de avaliar e como testar as partes"

00h 21' 42" - 00h 21' 50"

"And they tried to work that way, to make small deliveries, yet those small deliveries they made, we were not able to put it to test."

"E eles tentaram trabalhar essa forma, fazer entregas pequenas, só que essas entregas pequenas que eles faziam, a gente não conseguia colocar pra testar."

The need for more frequent meetings during the requirements gathering phase (SE)

00h 27' 47" - 00h 28' 33"

I just think the time gap between meetings, they were very long: it took a week for them [developers] to make a prototype, and her [PO] to receive a feedback,, right? So, in a week, they used to come, show the prototype and work on it, bring it back in the following week. This, I think, had to be a little more agile. And this is what they tried to do now in the second step [in 2018]. They themselves realized this: that it had a very gigantic interval (...)

“Eu só acho que o espaço das reuniões, eles eram muito demorados: até eles [developers] fazerem um protótipo, devolverem pra ela [PO], e isso levava uma semana, né? Então, vinha numa semana, mostrava, eles trabalhavam o protótipo, vinha na semana seguinte. Isso que eu acho que tinha que ser um pouco mais ágil. Que foi o que eles tentaram fazer agora na segunda etapa. Eles mesmos perceberam isso: que tinha um intervalo muito gigantesco (...)”

00h 46' 13" - 00h 46' 38"

“The way I see it is like this: we had a meeting on Monday, and my team, now I'm going to pass on to my team, my team will start developing. So, Tuesday, Wednesday, Thursday and Friday, the team is full of doubt, they are not able to produce. So, at this moment, we should have more contact with PO. At the requirements elicitation period. And I do not think that happened often, okay?”

“Mas só que eu acho assim, ó: a gente fez uma reunião na segunda-feira, e a minha equipe, agora eu vou passar pra minha equipe, minha equipe vai desenvolver. Então, terça, quarta, quinta e sexta, a equipe cheia de dúvida, ela não consegue produzir. Então, nesse momento aqui, tinha que ter um contato maior com o PO. No momento da coleta de requisitos. E eu acho que isso daqui não aconteceu muito, tá?”

Lack of development team experience (PO)

0h 19' 40" - 00h 20' 18"

“I thought it was time consuming. I'm not sure if it was too much or too little, but (...) the explanation I have is that they had to understand that business part, right? (...) Because it is an innovative process, which they do not know. Those who have had a traditional learning background do not know this type of evaluation, they link evaluation to tradicional examination.”

“eu achei que foi demorado. Não sei dizer demais, ou pouco, mas (...) a explicação que eu tenho é que eles tiveram que entender essa parte do negócio, né? (...) Porque é um processo inovador, que eles não conhecem. Quem teve uma formação tradicional de ensino não conhece esse tipo de avaliação, liga avaliação com prova.”

0h 23' 49" - 00h 24' 15"

“so, I had another system to show something already done, right? (...) now internal rule, I think it was more difficult for them to learn (...) and that part did not stay with the necessary time.”

“então eu tinha esse outro sistema pra mostrar já uma coisa feita, né? (...) agora regra interna, eu acho que foi mais difícil deles aprenderem (...) e essa parte não ficou com o tempo necessário.”

Lack of development team experience (SE)

01h 06' 57" - 01h 07' 38"

"If you asked me that question [how to identify system features from a natural language text] upon the time I left university in 1985, I would not be able to have that agility I have today. (...) and so, I imagine the [software house] with a lot of people working, and new people, who do not have so many experiences like that (...)"

"Se você fizesse essa pergunta pra mim [como identificar requisitos do sistema a partir de um texto em linguagem natural], lá quando eu saí da universidade, em 85, eu não ia conseguir ter essa agilidade que eu tenho hoje. (...) e aí eu imagino a [empresa de software] com um monte de pessoas trabalhando pra ela, e pessoas novas, que não têm tantas experiências assim (...)"

01h 27' 34" - 01h 28' 01"

"this is the difference from me to the software house staff, that at the time that asked to put the data, they placed it and, at the time that asked to retrieve the data, it was not found where it had to be. It is the experience of life that makes us so."

"essa é a diferença minha pro pessoal da [empresa de software], que na hora que pediu pra colocar os dados, colocou, na hora que pediu pra sair o dado, não estava onde tinha que estar. É a experiência da vida que faz a gente ser assim."

Using screen prototypes (PO)

00h 18' 11" - 00h 18' 31"

"First come the screens, then we make a final decision and next they implement (...) and it's a moment to review on this rules thing (...)"

"Primeiro vêm as telas, a gente bate o martelo e depois eles implementam (...) e é um momento de repassar essa coisa das regras (...)"

00h 36' 22" - 00h 36' 31"

"There is this quick response thing (...) It does not take so long for you to arrive [at the meeting] and see a screen, for example, that already gives a concrete material to whoever wants the system, don't you think?"

"Tem essa coisa da resposta rápida (...) Não demora tanto pra você chegar [na reunião] e ver uma tela, por exemplo, que isso dá uma concretude pra quem quer o sistema, né?"

Using screen prototypes (SE)

00h 27' 46" - 00h 29' 05"

“when they opted to build the prototype, I think it was a wise choice. I just think the time gap between meetings, they were very time consuming: until they make a prototype, return it to she [PO], it took a week, right?”

“quando eles optaram pela construção do protótipo, eu acho que foi uma escolha acertada. Eu só acho que o espaço das reuniões, eles eram muito demorados: até eles fazerem um protótipo, devolverem pra ela [PO], e isso levava uma semana, né?”

00h 33' 20" - 00h 33' 27"

“but it was more related to usability than to the functionality itself”

“mas era mais com relação a usabilidade que com relação à funcionalidade propriamente dita”

Not testable with real data set (SE)

00h 49' 01" - 00h 50' 19"

“tool to build the prototypes they used, next you had to program everything in the other language. So, maybe this prototype building tool, it should be a tool that will already be its tool after (...) with the possibility of the user putting data in it”

“ferramenta pra construir os protótipos que eles [desenvolvedores] usaram, aí depois você tinha que programar tudo na outra linguagem. Então, talvez essa construção do protótipo, ela ser numa ferramenta que já vai ser a sua ferramenta depois (...)com a possibilidade do usuário colocar dados nela”

Software that “handles the Medicine Programme needs” (PO)

00h 46' 47" - 00h 47' 02"

“to go back and revise that, only if we have a very big overhaul of the system, if not it will follow as it is (...) that he handles the Medicine Programme needs (...)”

“voltar e mexer nisso só se a gente tiver numa revisão muito grande do sistema pra reolhar isso, se não ele vai seguir como está mesmo (...) que ele está dando conta das necessidades do curso(...)”

Lack of openness for more effective collaboration, not taking advantage of SE meaningful prior experience (PO)

00h 41' 51" - 00h 42' 08"

"is that I think that since I know a little more, maybe I could have contributed a little more, maybe it was not my part of the job, you know? (...) the PO does not enter in that part. Why not, if the person has that possibility, why not go in too? "

"é que eu acho que pode ser que como eu conheço um pouco mais, talvez eu tivesse contribuído um pouco mais, talvez nao era a minha parte do latifúndio, entendeu? (...) o PO não entra nessa parte. Por que não, se a pessoa tem essa possibilidade, por que não entrar também?"

Lack of openness for more effective collaboration, not taking advantage of SE meaningful prior experience (SE)

00h 44' 58" - 00h 45' 07"

"for me, the software house had to have taken into consideration a little more, I was there, the experience of some things that I pointed out to them and they did not want to accept at that moment. "

"pra mim, a empresa de software tinha que ter levado em consideração um pouco mais, eu estava lá, a experiência de algumas coisas que eu apontava pra eles e eles não queriam aceitar naquele momento. "

Discontinuity and very delayed testing (PO)

00h 56' 05" - 00h 56' 37"

"And I think the discontinuity also helped to not making the whole process, right? Because I think we stopped at a time when I should had to test more (...) and I had not do it. Not even in 2015, it stopped, stopped and when they delivered in December 2017, in that general suffocation, I took 3 months to get the tests because before that, I could not do it. "

"E acho que a descontinuidade também acabou que ajudou a não fazer talvez o processo inteiro mesmo, né? Porque acho que a gente parou num momento em que eu teria que ter testado mais (...) e eu não fiz isso. Nem em 2015, parou, parou e quando eles entregaram em dezembro de 2017, naquele sufoco geral, eu demorei 3 meses pra conseguir fazer os testes, porque antes disso, eu não tinha condições de fazer "

Discontinuity and very delayed testing (SE)

00h 15' 48" - 00h 16' 37"

"we started with this in 2014, I think it was. By 2015, it's over. 2016 was stopped the whole year (...) I think that in the middle of the year of 2017 we resume, right? Then, we tried to put it in 2016 to run and we could not get teachers to come to the Medicine to use it. (...) So, when did we get to use the evaluation? Now, in the first half of 2018 "

“nós começamos com isso em 2014, acho que foi. Em 2015, terminou. 2016 ficou parado o ano inteiro (...) acho que no meio do ano de 2017 nós retomamos, né? Aí, nós tentamos colocar em 2016 pra rodar e nós não conseguimos professores que topassem lá na Medicina a usar. (...) Aí, então, quando que nós conseguimos usar a avaliação? Agora, no 1o semestre de 2018”

Transcription excerpts related do Cycle 2:

Phase for defining a common language between user and developer (PO)

00h 01' 14" - 00h 01' 41"

“We need to combine, right? Instrument means that, format that (...) active learning methodology for this project is this, because for another project it can be another thing, there are things that are more established, some less (...) and I think this process [ontologies], it can help developers understand more of the business domain.”

“A gente precisa combinar, né? Instrumento significa isso, formato aquilo (...) metodologia de ensino ativa pra este projeto é isso, porque pra outro pode ser uma coisa, tem coisas que estão mais estabelecidas, outras menos (...) e acho que esse processo [ontologias], ele pode ajudar o pessoal da área técnica entender mais do negócio.

00h 09' 09" - 00h 09' 20"

“There are more records of the explanation (...) there is a systematization of the conversations [ontologies], this is.”

“Tem mais registros da explicação (...) tem uma sistematização das conversas, [ontologias] é isso.”

Use of "records" that give visibility to the development phases and enable the PO collaboration on those phases (PO); and

Possibility of better result with less "rework" (PO)

00h 19' 23" - 00h 23' 46"

“that other part [ontologies], I find it to be something worth, which was something I missed having [cycle 1]. (...) Maybe here it is easier for me to see what was understood by the others [developers] and if I can help, improve or something that I did not remember talking before and I see here that it is missing, I might talk about it at that moment.”

“aquela outra parte [ontologias], achei que é uma coisa que vale à pena, que era uma coisa que eu sentia falta [no ciclo 1]. (...) Talvez aqui esteja mais fácil pra eu ver o que o outro [desenvolvedor] entendeu e se eu posso ajudar, melhorar ou uma coisa que eu não lembrei de falar lá atrás eu vejo aqui que está faltando, eu poderia falar nesse momento.”

00h 31' 52" - 00h 32' 04"

"Ah! I got it. This is the (...) the BDD? (...) And the boys [software house] there do not do that (...) And if I want to look I can not see this."

"Ah! Entendi. Esse aqui é o (...) o BDD? (...) E os meninos lá [empresa de software] não fazem isso.(...) E lá se eu quiser olhar eu não enxergo isso aqui."

00h 52' 37" - 00h 53' 07"

"Look, in general, what I noticed in the approach, it has something I thought it was fragile in the other method [cycle 1] which was the records issue, the possibility of having it with more details recorded, right? And having the record, to be able to validate, perhaps before doing, before going to the (...) implementation phase, right? (...) then I see advantages."

"Olha, de uma maneira geral, o que eu percebi é que esta proposta, ela tem uma coisa que eu achava frágil na outra que era a questão dos registros, da possibilidade de ter o detalhe mais registrado, né? E tendo o registro, poder validar, talvez antes de fazer, de partir pra fase de (...) implementação, né? (...) então eu vejo vantagens."

Use of "records" that facilitate the communication with the user (SE); and

Better communication with the developer, and consequent increase in productivity (SE)

01h 07' 51" - 01h 08' 13"

"so, when I see a model like this, where you have the feature, I want to (...) [reading ScrumOntoBDD BackLog structures] I think this is a model for you to follow, for you to do the requirements elicitation. And then you insert the user into that context and later, things are going to come out more or less the way they have to be. (...) I see advantage in this from here. It's the fact that you have a pattern and you use that pattern to do the requirements elicitation."

"aí, quando eu vejo um modelo como esse, então, que você tem lá a feature, a an, I want to (...) [lendo estruturas em inglês do BackLog] eu acho que isso é um modelo pra você seguir, pra você fazer requisito. E aí você insere o usuário nesse contexto e as coisas já vão sair mais ou menos do jeito que elas têm que ser depois. (...) Eu vejo vantagem nisso daqui. É o fato de você ter um padrão e você usar esse padrão pra fazer requisito."

01h 11' 20" - 01h 11' 43"

"And when you write that way [writing a text in natural language], it gives you the possibility of double interpretation. So when I use a model like this [ontologies], it has a more defined formalism, hence when I apply this formalism, it gives less possibility of double information (...) ambiguity. So I can extract the requirement easier.

"E quando você escreve dessa forma [redação de um texto], ela te dá possibilidade de dupla interpretação. Então, quando eu uso um modelo desse [ontologias], ele está mais com o formalismo definido, daí quando eu aplico esse formalismo, ele dá menos possibilidade de dupla informação (...) ambiguidade. Então eu consigo extrair o requisito de uma forma mais fácil."

Use of formal methodology to complement Scrum (SE)

01h 17' 25" - 01h 22' 12"

"The way I think is: Scrum is great for you to organize functions within the team (...) now, after you've split roles, you have to have this here to tell him how he does his activity. (...) So, I think this [BDD ontologies and scenarios], it complements Scrum for sure. And this is needed. Because based on my experience, there was no such formalization of how you will complement the other activities you have to do."

"Eu acho assim: o Scrum, ele é ótimo pra você organizar funções dentro da equipe (...) agora, depois que você dividiu papéis, você tem que ter isso aqui, ó, pra falar pra ele como é que ele faz a atividade dele. (...) Então, eu acho que isso [ontologias e cenários do BDD], ele complementa o Scrum sim. E precisa. Porque a experiência que eu tive, não existia essa formalização de como é que você vai complementar as outras atividades que você tem que fazer."

Does not use screen prototypes (PO); and Probably the usual POs will have understanding difficulties (PO)

01h 00' 09" - 01h 01' 03"

"I see the advantages and for me, now, the main disadvantage that I see is this thing is the time, the time necessary for me to see something concrete (...) although the approach here does not have, right? the screen prototypes, but it could, maybe if it is added here it may help (...) because for those who do not have this facility to think a lot here, to see, to read these lines of code, this does that and that does this, because not everyone will be patient."

"Vejo as vantagens e pra mim, agora, a principal desvantagem que eu vejo é essa coisa do tempo mesmo, do tempo pra eu ver alguma coisa concreta (...) apesar de que a proposta aqui não tem, né? os protótipos de tela, mas poderia, talvez se acrescentar isso aqui possa ajudar (...) porque pra quem não tem essa facilidade de pensar muito pra cá, de enxergar, ler esses montes de linhas, faz isso, faz aquilo, não é qualquer um que vai ter paciência."

Does not use screen prototypes (SE)

00h 44 07" - 00h 44' 52"

"The prototype I would take, because I saw that it was something that worked (...) because it was a palpable means of discussing (...) because otherwise everything stays only in the head, everything in the diagrams. So, the prototype even though it was navigable, it helped to give an idea (...)"

"O protótipo eu levaria, porque eu vi que foi uma coisa que deu resultado (...) porque era um meio palpável de se discutir (...) porque senão fica tudo muito na cabeça, tudo muito nos diagramas. Então, o protótipo mesmo ele sendo navegável, ele deu pra dar uma ideia (...)"

Probably the usual POs will have understanding difficulties (SE)

01h 35 29" - 01h 36' 03"

"I do not guarantee that this will succeed with that other user who wants to speak quickly and get you out of the meeting because he has a lot of other things to do (...) If all POs will understand it, I have my doubts"

"Eu não garanto que isso vai ter sucesso com aquele outro usuário que quer te falar rapidamente e vazar da reunião porque ele tem um monte de outras coisas pra fazer. (...) Se todos os POs vão entender isso, tenho dúvidas"

Need for an additional actor, the Ontology Engineer, and Likely an increase on the system cost (PO); and

Likely increase in system development time (PO)

01h 00' 12" - 01h 02' 06"

"and one more thing I thought is that the moment I put in another development role [OE], I'm going to increase the cost and time (...) but it's an increase in cost and time that pays off because I will have a better result, less rework in the process? (...) Could be. But this I cannot tell you. The impression I have is that it increases the cost and time, but considering the advantages, it might be worth it, right?"

"e mais uma coisa que eu pensei é que na hora que eu coloco mais uma figura de desenvolvimento [engenheiro de ontologias], eu vou aumentar o custo e tempo (...) mas é um aumento de custo e tempo que compensa porque eu vou ter um melhor resultado, menos retrabalho no processo? (...) pode ser. Mas isso eu não tenho como dizer pra você. A impressão que eu tenho é que aumenta o custo e tempo, mas considerando as vantagens, pode ser que ele valha à pena, né?"

Likely reduction of "rework" (SE); and

Probably better cost/benefit ratio (SE)

01h 36' 58" - 01h 37' 48"

"everyone thinks that formalism consumes time. But she [PO] can see the time she is spending now to do all the tests, that things were done with little bugs that maybe would be solved, because a bug is not a lack of understanding, it is inexperience, and maybe they'd be resolved here, if at this time we had said a little more (...) if you have formalism (...) we have been testing since January, we are already in August! (...) it's been too long and it is still full of problems"

"todo mundo acha que o formalismo toma tempo. Mas ela [PO] está vendo o tempo que está levando agora pra fazer todos os testes, que as coisas foram feitas com bugzinhos que talvez seriam solucionados, porque bug não é falta de compreensão, é inexperiência, talvez seriam solucionados aqui, se nesse momento a gente tivesse dito um pouco mais (...) se tem um formalismo (...) nós estamos testando desde janeiro, nós já estamos em agosto! (...) é muito tempo e ainda está lotado de problema"

01h 41' 54" - 01h 42' 00"

"So, do you consume a little more time? Yes, you do. But I assure you that you will have less error"

"Então, consome um pouco mais de tempo? Consome sim. Só que eu garanto pra você que você vai ter menos erro"

4.3.4 Specifying Learning

The objective of this section is to report the learning resulting from this experiment, mainly concerning the research questions RQ₁ and RQ₂ that motivated the development of this MSc's project. Thus, we will present our conclusions based on the analysis main results described in the precedent section, and summarized in Tables 10 and 11.

As can be observed in these tables, the PO comments and evaluations are mainly focused on aspects and issues related to the development process of a product that "responds" to the user demands. On the other hand, the SE comments and evaluations are mainly focused on aspects and issues related to the software development methodologies employed on Cycle 1 and on Cycle 2.

Product Owner (PO)		Software Engineer (SE)	
Focus on the application development		Focus on the development methodology	
Advantages	Disadvantages	Advantages	Disadvantages
Agility due to weekly meetings	Lack of more systematic "records" that could give more visibility to the different development phases	Using agile software development methodologies like Scrum	Scrum distributes roles and activities to team members, but does not explain how to implement these activities Lack of a formal methodology that facilitates the communication with the user and with the developer
			Division into non-testable sub-tasks throughout the process The need for more frequent meetings during the requirements gathering phase
Using screen prototypes		Using screen prototypes	Not testable with real data set
Software that "handles the Medicine Programme needs"	Lack of development team experience		Lack of development team experience
	Lack of openness for more effective collaboration, not taking advantage of PO meaningful prior experience		Lack of openness for more effective collaboration, not taking advantage of SE meaningful prior experience
	Discontinuity and very delayed testing		Discontinuity and very delayed testing

Table 10: Main results of the Evaluating phase of Cycle 1

Product Owner (PO)		Software Engineer (SE)	
Focus on the application development		Focus on the development methodology	
Advantages	Disadvantages	Advantages	Disadvantages
Phase for defining a common language between user and developer		Use of formal methodology to complement Scrum	Longer development process
Use of "records" that give visibility to the development phases and enable the PO collaboration on those phases	Does not use screen prototypes Probably the usual POs will have understanding difficulties	Use of "records" that facilitate the communication with the user	Does not use screen prototypes Probably the usual POs will have understanding difficulties
	Need for an additional actor, the Ontology Engineer, and likely an increase on the system cost	Better communication with the developer, and consequent increase in productivity	
	Likely increase in system development time	Likely reduction of "rework" (reparação de erros)	
Possibility of better result with less "rework"		Probably better cost/benefit ratio	

Table 11: Main results of the Evaluating phase of Cycle 2

The results shown in Table 10 indicate that, despite the focus difference, the difficulties identified on Cycle 1 by both interviewees converge. These results corroborate with our assumption that the exclusive use of Scrum method in Cycle 1 would reveal communication issues between the PO and DT, insofar as information is lost in EAR and BPB activities. In the first one, due to the interaction between PO and DT almost always been in natural language and through texts, and since all information extracted from the user stories are described exclusively using BPMT. In the second one, due to the mapping of the informally specified requirements to PB items, which are also usually informally or semi-formally described.

Others significant convergences indicated by the results in Table 10 concerns the very long time of application development, and the lack of DT openness in

receiving a more effective collaboration from the PO and SE, both with significant prior experience in their respective roles. Even though the interviewees agree that the main cause of significant delays in the PI releases was due to most tests being insufficient and non-continuous throughout the software development, they pointed out other factors that should be taken into account. The PO emphasizes the complexity of the requirements related to the UFSCar Medicine Programme evaluation system as a factor to be considered, but for the SE the adopted method and the DT lack of experience are the other relevant factors to be considered.

Based on Table 10 results and the considerations above, our main learning from Cycle 1 is: there is not enough evidence that our research questions RQ₁ and RQ₂, input of this AR experiment, were satisfactorily answered using Scrum. With the exception of the screen prototypes, which both interviewees pointed out to be a very productive resource in the communication between PO and DT, all other comments reinforced the problems related to RQ₁ and RQ₂. This learning motivated us to verify if those same problems could be better solved using ScrumOntoBDD. So, we used them to feed the Diagnosing phase of Cycle 2, whose main objective is to verify if the interviewees had experienced the same problems reported in Cycle 1, and if not how ScrumOntoBDD helped to avoid or solve them.

The results shown in Table 11 indicate that, despite the focus difference, there are convergences in the interviewees' views regarding the possibilities created by ScrumOntoBDD to overcome the problems verified during Cycle 1. In this sense, the ScrumOntoBDD characteristics that are considered to be the most advantageous are: **(1)** the ability to define a common language between users and developers by employing ontologies and BDD; **(2)** the generation of "records" that give visibility to each phase of software development, which can also improve the communication and participation of PO within the DT; and **(3)** the reduction of the total time spent repairing software bugs. Another convergence to be pointed out is the need to introduce into ScrumOntoBDD the screen prototypes usage, since it was considered a major advantage in Cycle 1 by both interviewees.

The results in Table 11 show also convergences on some problems pointed out by the interviewees about ScrumOntoBDD usage. The first one concerns possible

difficulties, to be faced by less experienced or less available POs, for understanding the formalism involved in this approach since its first activity. The second one is the longer development time required for creating the ontologies, and the higher cost due to the insertion of a new actor, i.e., the Ontology Engineer (OE). Nevertheless, both interviewees point out the potential of ScrumOntoBDD to obtain better results in terms of cost-benefit. The SE insists on the potential significant reduction of time spent in the software development process as a whole, due to the smaller number of errors she believes this approach would produce.

Based on Table 11 results and the considerations above, our main learning from Cycle 2 is: there is evidence that our research hypotheses H_1 and H_2 , i.e., combining BDD with Scrum can improve the communication between PO and developers, and employing ontologies can eliminate the ambiguities intrinsic of using natural languages to report user stories, are both held when employing ScrumOntoBDD for agile software development.

4.4 Final Considerations

In this chapter we presented an experimental analysis of ScrumOntoBDD for agile software development, whose main objective was to evaluate our two hypotheses when employing this approach. The research used for the experimental analysis was Action Research for two reasons: it is a cyclical process guided by the collaborative and continuous learning principles; and the AR allows the cooperative and participative involvement between the researcher and the actors of the experiment.

The experiment was conducted with the participation of the same PO and the same PhD's student, who were involved in the EAMS-CBALM development by the software house, for being able to evaluate our hypotheses. A usage scenario was chosen in the Education domain, inside the context of the UFSCar Medicine Programme, and supported by EAM-CBALM, which has similarities to the one employed to illustrate ScrumOntoBDD. Therefore, most of the ontologies already created and validated were reused. Nevertheless, the new usage scenario was more

complex, and more often requested by system's users than the one used to illustrate the approach.

In order to have PO and SE independent views of the experiment, it was executed it twice: first involving PO and researcher; and second involving SE and researcher. The AR phases were followed and its results were analysed with the supervision of an AR expert. Each phase was detailed and illustrated with excerpts of the material used and produced during the experiment. In the evaluation phase, the semi-structured interview model was chosen since it follows a pre-defined script of subjects to be addressed with the interviewees, instead of a sequence of issues of questionnaire type. Both interviews were audio-taped and fully transcribed in Portuguese. The transcriptions were analysed, their most relevant excerpts were selected and summarized into tables, allowing a better understanding of the learning results from the experiment cycles.

The conclusions based on the analysis of the experiment main results indicated that ScrumOntoBDD allows for both, improving quite a lot the communication between PO and developers (**H₁**), and eliminating quite a lot the ambiguities intrinsic of using natural languages to report user stories (**H₂**).

The following chapter will present some related works to the one described in this MSc's dissertation, splitting them in two sections: the related works that employ BDD to improve software development; and the related works that employ Ontology to improve software development.

Chapter 5

RELATED WORKS

This chapter presents some related works to the one described in this MSc's dissertation, splitting them in two sections according to the two systematic literature reviews carried out during the development of this MSc's project. Section 5.1 presents the most related works to ours that employ BDD to improve software development; Section 5.2 presents the most related works to ours that employ Ontology to improve software development and Section 5.3 has the final considerations.

5.1 BDD Related Works

The rising popularity of BDD has spurred research to apply it into a variety of different domains (Diepenbeck, 2015). Since currently available BDD tools give little or no support to the planning phase, most of the BDD related works, we found in our first systematic literature review, define a specific ubiquitous language for a given application domain.

The paper "Modeling test cases in bpmn for behavior-driven development" (Lubke, 2016) presents a platform for integrating systems responsible for land registration in Switzerland. The goal is to reduce process execution time between systems and also the communication time between POs. In order to model executable integration processes between various systems and to develop test cases to validate these processes, the authors used BDD and Business Process Model

Notation (BPMN) as the ubiquitous language for the test case models (scenarios) construction. The BPMN was chosen because this language is known by the POs. By combining BDD with BPMN scenarios, the authors found improvements in communication between developers, users and investors, which contributed significantly to the more agile development of the platform.

The paper “Testing of web services using behavior-driven development” (Oruç, 2016) proposes a tool to facilitate the creation of scenarios for web services test. This tool uses BDD in conjunction with the ubiquitous language Gherkin (Wynne, 2012) for dynamically generating test scripts. These scripts are run in JMeter, a test tool for analysing and measuring the performance of web applications (JMeter, 2016). The authors claim to achieve two benefits with this tool: because Gherkin is a domain-specific language, it allows any domain expert to create and run web services tests even without software knowledge; and developers do not need to write unit tests manually since JMeter automates testing.

The paper “Testing prototypes and final user interfaces through an ontological perspective for behavior-driven development” (Silva, 2016) proposes an approach based on BDD to support the automated assessment of artefacts along the development process of interactive systems. A formal ontology model is defined for describing concepts used by platforms, models and artefacts that compose the design of interactive systems, allowing in this way a wide description of User Interface (UI) elements and its behaviours to support testing activities. In addition, the approach proposes improvements in how teams must write requirements for testing purposes. Once described in the ontology, the behaviours can be reused freely to write new scenarios in natural language, providing test automation in BDD and decreasing manual coding.

The paper “Assisted behavior driven development using natural language processing” (Soeken, 2012) presents a methodology to assist developers carrying out the BDD steps. This methodology proposes a design flow where the developer engages in a dialog with a computer program in an interactive way. This dialog contains the user history, and this program processes each spoken sentence and generates the step definitions and code blocks (classes, attributes and methods) of

each user story scenario. Some natural language processing tools are explored and a case study illustrates the application. Rather than going manually through the established BDD steps, this methodology suggests some scenario skeletons facilitating tests refinement and implementation.

The main difference between our work and those presented in this section is that, in addition to applying BDD in software development, our work also exposes the benefits of using BDD in combination with Scrum. Moreover, our case studies were developed for the Education domain, more specifically for courses based on active learning methodologies, and we didn't find in our systematic literature review this combination of BDD and Scrum for information system development in this domain.

5.2 Ontology Related Works

Ontologies have been used in many areas of Computer Science, and in the Software Engineering area most of the Ontology related works we found in our second systematic literature review, propose a process, or an approach, or another way to combine ontologies with some agile software methodology in order to improve software development.

The paper "OntoSoft Process: Towards an agile process for ontology-based software" (Machado, 2016) proposes an agile process that associates practices of Software Engineering, Ontology Engineering and Scrum for improving the collaboration between software and ontology engineers. This process provides a set of guidelines for defining activities, tasks, roles, and artefacts to develop ontology-based software. OntoSoft was applied for developing an ontology-based application to map and recommend real estates. This paper synthesizes the main results obtained in the development of a PhD's project, and its complete description can be found in (Machado, 2017), a PhD's thesis written in Portuguese.

The paper "Multi-Agent System for intelligent Scrum project management" (Lin, 2015) presents an approach to help team members perform more efficiently their daily tasks according to a specific process. This approach is based on the K-

CRIO ontology for business processes modeling and on a multi-agent system for providing intelligent assistance to workers.

The paper “Improving agile requirements: the Quality User Story framework and tool” (Lucassen, 2016) proposes the QUS framework for ensuring the quality of agile requirements expressed as user stories. QUS contains 13 criteria that determine the quality of user stories in terms of syntax, semantics, and pragmatics. Based on QUS, the Automatic Quality User Story Artisan (AQUSA) tool was built to detect QUS quality criteria violations, and to improve user stories.

The paper “A Behavior-Based Ontology for Supporting Automated Assessment of Interactive Systems” (Silva, 2017) introduces an ontological model to support scenario description and to test functional requirements of interactive systems. This model was developed based on BDD principles, describing user behaviours when interacting with User Interface (UI) elements in a scenario-based approach. Once described in the ontology, behaviours can be freely reused to write new scenarios in natural language, providing test automation. A case study is presented for the flight tickets e-commerce domain, where ontology-based tools were used to support the assessment of evolutionary prototypes and final UIs.

5.3 Final Considerations

All the works presented in this section employ ontologies for specific purposes: for boosting the collaboration among software engineers and ontology engineers, for modeling the Scrum development process, for extracting semantic information to improve user stories, and for supporting test automation. The main difference between these works and ours is that we use ontologies in a broader context, starting from a reference ontology for a given domain, and gradually specializing it to be used in the agile software development for that domain.

Last but not least, our work proposes an integrated approach that combines three disciplines, i.e., Scrum, Ontology and BDD, for improving agile software

Chapter 5: Related Works

development, and to the best of our knowledge this combination has not been addressed before.

The following chapter will present this MSc's dissertation concluding remarks, discussing its main contributions and limitations, and giving directions for future works.

Chapter 6

CONCLUSION

This final chapter presents our concluding remarks of this work, discussing its main contributions and limitations, and giving directions for future works.

6.1 Contributions and Limitations

The main motivation for doing this research was the following problems we observe during the EAMS-CBALM development using Scrum: it was quite often necessary to redefine some system behaviour scenarios, and consequently the PB items, due to misunderstanding of the stories reported by the PO; and the definition of test suites was cumbersome, resulting in test suites that were incomplete or did not comply with the system requirements.

These problems raised the following research questions: how to improve the communication between PO and developers (RQ₁); and how to eliminate the ambiguities intrinsic of using natural languages to report user stories (RQ₂). Based on these questions, we have established two main hypotheses: combining BDD with Scrum can improve quite a lot this communication (H₁); and employing ontologies can eliminate quite a lot these ambiguities (H₂).

The first contribution of this MSc's project was the development of a case study in the EAMS-CBALM context for verifying H₁. It was developed by employing a

ubiquitous language for the Education domain together with BDD scenarios and acceptance tests, for allowing the PO to follow and properly communicate with the developers throughout the development process. This work was reported in (Souza, 2017).

The second contribution of this MSc's project was the development of another case study also in the EAMS-CBALM context for verifying H₂. It was developed by employing domain ontologies to describe the UFSCar Medicine Programme, in order to reduce the ambiguities caused by using a natural language as a ubiquitous language. This work was reported in (Souza, 2018).

The main contribution of this MSc's project was the development of ScrumOntoBDD approach, which combines Scrum, Ontology and BDD, for joining the two precedent works, and for assuring both hypotheses. An experimental analysis of ScrumOntoBDD was carry out using Action Research, in order to get evidences that the two hypotheses hold when employing this approach. This work was reported in this MSc's dissertation.

The main difficulty we found during the development of this MSc's project was related to ScrumOntoBDD validation. Our hypotheses deals with the communication between POs and developers, and all works we have done were in the EAMS-CBALM context, a product developed by a software house. Therefore, for a concluding verification of these hypotheses employing our approach, it would be necessary to develop an experiment involving the complete EAMS-CBALM development team, and that was not possible. So, the ScrumOntoBDD evaluation we performed was limited to find evidences that our hypotheses hold when employing this approach.

6.2 Future Works

Immediately, we should incorporate the improvements in ScrumOntoBDD approach suggested by the experiment participants. For example, the inclusion in the Defining Sprint Backlog activity of the Screen Prototype Generation task related to

the SelectedBDDUSS item of the Backlog, so these screen prototypes would be validated by the PO and, if needed, this item could be adapted before output it to the next activity, i.e., Executing Sprint Backlog.

We intended to write a paper describing the whole work done in this MSc's project, which was reported in this MSc's dissertation. This paper should be submitted to an international journal evaluated (Qualis) as A₁, or A₂ in the Computer Science area by the Coordination for the Improvement of Higher Education Personnel (CAPES).

The next work should be a quantitative study of the ScrumOntoBDD approach through an experimental methodology as the one presented in (Wohlin, 2000). This study must involve two development teams that will separately develop the same application twice: in the first, a team will employ Scrum, and the other will employ ScrumOntoBDD; and in the second, these approaches will be reversed. Data such as the time spent on each development of the application should be collected for further analysis.

Finally, we believe this project can be extended in order to transform the ScrumOntoBDD approach into a process, by refining its activities and the involved tasks, thus providing this way more detailed guidelines for development teams.

REFERENCES

Alatrish, E., 2013. "Comparison Some of Ontology Editors". International Scientific Journal of Management Information Systems, Vol. 8, No. 2, pp. 18-24.

Beck, K. et al., 2001. "Manifesto for Agile Software Development". www.agilemanifesto.org/ (accessed August 27, 2018).

Beck, K., 2002. "Test Driven Development: By Example". Addison-Wesley, 240 pgs.

Beck, K., 2012. "Extreme Programming Explained: Embrace Change". Second edition, Addison-Wesley, 189 pgs.

Berners-Lee, T., 2009. "The Semantic Web as a language of logic". Available at <https://www.w3.org/DesignIssues/Logic.html#Crawf90> (accessed August 27, 2018).

Bray, T. et al, 2008. "Extensible Markup Language (XML) 1.0 (Fifth Edition)". W3C Recommendation. Available at <https://www.w3.org/TR/xml/> (accessed August 27, 2018).

Brickley, D.; Guha, R. V., 2014. "RDF Schema 1.1". W3C Recommendation. Available at <https://www.w3.org/TR/rdf-schema/> (accessed August 27, 2018).

Carlisle, C.; Calman, L.; Ibbotson, T., 2009. "Practice-based learning: The role of practice education facilitators in supporting mentors". Nurse Education Today, Vol. 29, No 7, pp. 715-721.

Chauhan, R., 2016. "Fusion of Agile-Scrum to BDD". Available at <http://docplayer.net/28500425-Fusion-of-agile-scrum-to-bdd.html> (accessed August 27, 2018).

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

Chen, G. D.; Chang, C. K.; Wang, C.Y., 2008. "Ubiquitous learning website: Scaffold learners by mobile devices with information-aware techniques". *Computers & Education*, Vol. 50, No 1, pp. 77–90.

Cockburn, A, 2004. "Crystal Clear: A Human-Powered Methodology for Small Teams". Addison-Wesley, 336 pgs.

Connolly, D. et al, 2001. "DAML+OIL (March 2001) Reference Description". W3C Note. Available at <https://www.w3.org/TR/daml+oil-reference> (accessed August 27, 2018).

Cucumber, 2014. "Cucumber". Available at <http://cukes.info/> (accessed August 27, 2018).

Cycorp, 2014. "Cyc: Logical Reasoning with the Worlds's Largest Knowledge Base". Available at <http://www.cyc.com/> (accessed August 27, 2018).

Dingsoyr, T.; Lassenius, C. Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and Software Technology*, v. 77, p. 56-60, 2016.

Diepenbeck, M. et al, 2014. "Behavior Driven Development for Tests and Verification". *Tests and Proofs, Lectures Notes in Computer Science (LNCS)*, Springer International Publishing AG, Vol. 8570, pp. 61-77.

E-learning, 2018. "Learning Management Systems: Find, choose and compare the eLearning Industry's Top LMS Software". eLearning INDUSTRY. Available at <https://elearningindustry.com/directory/software-categories/learning-management-systems> (accessed August 27, 2018).

EI-Fakih, K. et al, 2016. "Distinguishing Extended Finite State Machine Configurations Using Predicate Abstraction". *Journal of Software Engineering Research and Development*, Vol. 4, No 1, 26 pgs.

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

Evans, E., 2003. "Domain-Driven Design: Tackling Complexity in the Heart of Software". Addison-Wesley Professional, 529 pgs.

Fensel, D. et al, 2000. "OIL in a Nutshell". In Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW'00), 16 pgs.

Forte, M. et al, 2013. "A Ubiquitous Reflective E-Portfolio Architecture". International Journal of Medical Informatics, Vol. 82, No. 11, pp. 1111-1122.

Gandon, F.; Schreiber, G., 2014. "RDF 1.1 XML Syntax". W3C Recommendation. Available at <https://www.w3.org/TR/rdf-syntax-grammar/> (accessed August 27, 2018).

Gangemi, A. et al, 2006. "Modelling ontology evaluation and validation". Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 4011, pp. 140-154.

Genesereth, M. R., 2004. "Knowledge Interchange Format (KIF)". Draft proposed American National Standard (dpANS), NCITS.T2/98-004, Stanford University. Available at <http://logic.stanford.edu/kif/dpans.html> (accessed August 27, 2018).

Gomaa, H., 2004. "Designing software product lines with UML". [S.I.]:Addison-Wesley Boston (USA).

Griss, M.; Favaro, J.; D'Alessandro, M., 1998. "Integrating feature modeling with the RSEB". In IEEE Proceedings of the Fifth International Conference on Software Reuse, pp. 76–85.

Gruber, T. R., 1995. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing". International Journal Human-Computer Studies, Vol. 43, Issues 5-6, pp. 907-928. Available at <http://tomgruber.org/writing/onto-design.pdf> (accessed August 27, 2018).

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

Gruber, T. R., 2008. "Ontology". Encyclopedia of Database Systems, Springer-Verlag, 04 pgs. Available at <http://tomgruber.org/writing/ontology-in-encyclopedia-of-dbs.pdf> (accessed August 27, 2018).

Guarino, N, 1998. "Formal Ontology and Information Systems". In Proceedings of 1st International Conference on Formal Ontology in Information Systems (FOIS'98), N. Guarino (ed.), Frontiers in Artificial Intelligence and Applications, IOS Press, Vol. 46, pp. 3-15.

Guri-Rosenblit, S., 2005. "'Distance education' and 'e-learning': Not the same thing". Higher Education, Springer International Publishing AG, Vol. 49, Issue 4, pp. 467-493.

Harith, A.; Kieron, O.; Nigel, S., 2005. "Common Features of Killer Apps: A Comparison with Protégé". In Proceedings of the 8th International Protégé Conference, 04 pgs. Available at <https://eprints.soton.ac.uk/260989/1/protege05-Alani.pdf> (accessed August 27, 2018).

Highsmith, J. A., 2000. "Adaptive Software Development: A Collaborative Approach to Managing Complex Systems". Dorset House Publishing, 358 pgs.

Horrige, M, 2007. "A practical guide to building owl ontologies using Protégé 4.0 and CODE tools". Edition 1.1, University of Maryland.

Järvinen, P., 2007. "Action Research is Similar to Design Science". Quality & Quantity International Journal of Methodology, Springer, Vol. 41, Issue 1, pp. 37-54.

JBehave, 2015. "JBehave". Available at <http://jbehave.org/> (accessed August 27, 2018).

JUnit, 2016. "JUnit". Available at <http://junit.org/junit4/> (accessed August 27, 2018).

JMeter, 2016. "JMeter graphical server performance testing tool". Available at <http://jmeter.apache.org/> (accessed August 27, 2018).

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

Kifer, M.; Lausen, G.; Wu, J., 1995. "Logical Foundations of Object-Oriented and Frame-Based Languages". *Journal of the ACM*, Vol. 42, No 4, pp. 741–843.

KMI, 2004. "Apollo". Knowledge Media Institute (KMI), The Open University. Available at <http://apollo.open.ac.uk/index.html> (accessed August 27, 2018).

Koskela, L., 2008. "Test Driven: TDD and Acceptance TDD for Java Developers". Manning Publications Co., 513 pgs.

Lewin, K., 1988. "Group decision and social change". *The Action Research Reader*, S. Kemmis (ed.), Deakin University Press, Victoria (Australia), pp. 47–56.

Lin, Y. et al, 2015. "Multi-Agent System for intelligent Scrum project management". *Integrated Computer-Aided Engineering*, IOS Press, Vol. 22, No 3, pp. 281-296.

Lubke, D.; Van Lessen, T., 2016. "Modeling test cases in bpmn for behavior-driven development". *IEEE Software*, Vol. 33, No 5, pp. 15–21.

Lucassen, G. et al, 2016. "Improving agile requirements: the Quality User Story framework and tool". *Requirements Engineering*, Springer International Publishing AG, Vol. 21, No 3, pp. 283-403.

Luke, S.; Hein, J., 2000. "SHOE 1.01 Proposed Specication". Available at <https://www.cs.umd.edu/projects/plus/SHOE/spec.html> (accessed August 27, 2018).

MacGregor, R., 1991. "Using a Description Classifier to Enhance Deductive Inference". In *Proceedings of the Seventh IEEE Conference on AI Application*, pp. 93-97.

Machado, J. B. et al, 2016. "OntoSoft Process: Towards an agile process for ontology-based software". *Proceedings of 49th Hawaii International Conference on System Sciences*, IEEE Computer Society, pp. 5813-5822.

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

Machado, J. B., 2017. "OntoSoft: um processo de desenvolvimento ágil para software baseado em ontologia". PhD's thesis, Graduate Program in Computer Science and Computational Mathematics (PPG-CCMC) of University of São Paulo, Brazil, 195 pgs.

Mealy, G. H., 1967. "Another Look at Data". Proceedings of the Fall Joint Computer Conference, pp. 525-534. Available at <https://www.computer.org/csdl/proceedings/afips/1967/5070/00/50700525.pdf> (accessed August 27, 2018).

Mejía-Gutiérrez, R.; ;Carvajal-Arango, R., 2017. "Design Verification through virtual prototyping techniques based on Systems Engineering". Research in Engineering Design, Springer London, Vol. 28, Issue 4, pp. 477–494.

MIND SWAP, 2004. "Semantic Web Ontology Overview and Perusal (SWOOP)". Maryland Information and Network Laboratory (MIND), Semantic Web and Agents Project (SWAP), University of Maryland. Available at <https://github.com/ronwalf/swoop> (accessed August 27, 2018).

Moodle, 2018. "Moodle: Community driven globally supported". Moodle Partners. Available at <https://moodle.org/?lang=en> (accessed August 27, 2018).

Motta, E., 1998. "An Overview of the OCML Modeling Language". In Proceedings of 8th Workshop on Knowledge Engineering Methods & Languages (KEML'98), 19 pgs.

MSpec, 2008. "MSpec". Available at <https://github.com/machine/machine.specifications> (accessed August 27, 2018).

Musen, M. A., 2015. "The Protégé project: A look back and a look forward". AI Matters, Association of Computing Machinery Specific Interest Group in Artificial Intelligence , Vol. 1, Issue 4, pp. 4-12, doi.org/10.1145/2557001.25757003. Available at <https://protege.stanford.edu/> (accessed August 27, 2018).

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

NBehave, 2011. "NBehave". Available at <https://github.com/nbehave/> (accessed January 29, 2017).

North, D., 2006. "Introducing BDD". Dan North & Associates. Available at <http://dannorth.net/introducing-bdd> (accessed August 27, 2018).

North, D., 2007. "What's in a Story?". Dan North & Associates. Available at <https://dannorth.net/whats-in-a-story> (accessed August 27, 2018).

Nuzhat, H., 2011. "Empirical research consolidation: a generic overview and a classification scheme for methods". *Quality & Quantity International Journal of Methodology*, Springer, Vol. 47, Issue 1, pp. 383-410.

OMG, 2011. "About the Business Process Model and Notation Specification 2.0". Object Management Group. Available at: <https://www.omg.org/spec/BPMN/2.0> (accessed August 27, 2018).

OMG, 2017. "About the Unified Modeling Language Specification Version 2.5.1". Object Management Group. Available at: <https://www.omg.org/spec/UML> (accessed August 27, 2018).

Oruç, A. F.; Ovatman, T., 2016. "Testing of web services using behavior-driven development". In *CLOSER 2016 - Proceedings of the 6th International Conference on Cloud Computing and Services Science*, Vol. 2, pp. 85–92, Rome (Italy), April 23-25, 2016.

Palmer, S. R.; Felsing, J. M., 2002. "A Practical Guide to Feature-Driven Development". Prentice Hall, 271 pgs.

Prieto-Diaz, R.; Arango, G, 1991. "Domain analysis and software systems modeling". [S.I.]: IEEE Computer Society.

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

Protégé, 2017. “A free, open-source ontology editor and framework for building intelligent systems”. Available at <https://protege.stanford.edu/> (accessed August 27, 2018).

Rhem, J., 1998. “Problem Based Learning an Introduction”. The National Teaching and Learning Forum, Vol. 8, No 1, 07 pgs. Available at <http://www1.udel.edu/pbl/dec-june2006/supplemental/NTLF-PBL-introduction.pdf> (accessed August 27, 2018).

RSpec, 2016. “RSpec”. Available at <http://rspec.info/> (accessed August 27, 2018).

Ross, D. T., 1977. “Structured analysis (sa): A language for communicating ideas. IEEE Transactions on Software Engineering, IEEE Press, Vol. 3, pp. 16-34.

Rubin, K. S., 2012. “Essential Scum: A Practical Guide to the Most Popular Agile Process”. Addison-Wesley, 482 pgs.

Santos, H. F. et al, 2016. “Augmented Reality Approach for Knowledge Visualization and Production (ARAKVP) in Educational and Academic Management System for Courses Based on Active Learning Methodologies (EAMS–CBALM)”. In Proceedings of 13th International Conference on Information Technology: New Generations (ITNG 2016), Advances in Intelligent Systems and Computing, Springer International Publishing AG, Vol. 448, pp. 1113-1123.

Schwaber, K.; Sutherland, J., 2017. “The Scrum Guide™ - The Definitive Guide to Scrum: The Rules of the Game”. Scrum.Org and ScrumInc, 19 pgs. Available at <https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100> (accessed August 27, 2018).

Silva, T.; Hak, J.-L.; Winckler, M., 2016. “Testing prototypes and final user interfaces through an ontological perspective for behavior-driven development”. Lecture Notes in Computer Science, 9856, pp. 86–107.

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

Silva, T. et al, 2017. "A Behavior-Based Ontology for Supporting Automated Assessment of Interactive Systems". Proceedings of 11th International Conference on Semantic Computing, IEEE Computer Society, pp. 250-257.

Smith, B.; Welty, C., 2001. "Ontology: Towards a New Synthesis". Second International Conference on Formal Ontology and Information Systems, 07 pgs. Available at <http://mba.eci.ufmg.br/downloads/recol/piii-foreword.pdf> (accessed August 27, 2018).

Soeken, M.; Wille, R.; Drechsler, R., 2012. "Assisted behavior driven development using natural language processing". Lecture Notes in Computer Science, 7304, pp. 269–287.

Solis, C.; Wang, X., 2011. "A Study of the Characteristics of Behaviour Driven Development". In Proceedings of SEAA 2011: 37th EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE Computer Society, ISBN 978-0769544885, pp. 383-387, , Oulu (Finland), August 30-September 02, 2011.

Souza, P. L. et al, 2017. "Combining Behaviour-Driven Development with Scrum for Software Development in the Education Domain". In Proceedings of 19th International Conference on Enterprise Information Systems (ICEIS 2017), SCITEPRESS – Science and Technology Publications Lda, Vol. 2, pp. 449-458.

Souza, P. L. et al, 2018. "Improving Agile Software Development with Domain Ontologies". In Proceedings of 15th International Conference on Information Technology: New Generations (ITNG 2018), Advances in Intelligent Systems and Computing, Springer International Publishing AG, Vol. 738, Chapter 37, pp. 267-274.

SpecFlow, 2016. "SpecFlow". Available at <http://www.specflow.org/> (accessed August 27, 2018).

Stapleton, J., 2003. "DSDM: Business Focused Development". Addison-Wesley, 239 pgs.

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

StoryQ, 2010. "StoryQ". Available at <http://storyq.codeplex.com/> (accessed August 27, 2018).

Su, X.; Ilebrikke, L., 2002. "A Comparative Study of Ontology Languages and Tools". In Proceedings of 14th International Conference on Advanced Information Systems Engineering (CAiSE'02), Lectures Notes in Computer Science (LNCS), Springer International Publishing AG, Vol. 2348, pp. 761–765.

TllyFy, 2018. "9 Best Business Process Modeling Techniques (With Examples)". TllyFy. Available at <https://tallyfy.com/business-process-modeling-techniques/> (accessed August 27, 2018).

TopQuadrant, 2011. "TopBraid Composer Free Edition (FE)". TopQuadrant, Inc. Available at <https://www.topquadrant.com/topbraid-5-2-data-migration/> (accessed August 27, 2018).

UFSCar, 2007. "Curso de Medicina - CCBS Projeto Político Pedagógico". Medicina UFSCar, 139 pgs. Available at: <http://www.prograd.ufscar.br/cursos/cursos-oferecidos-1/medicina/medicina-projeto-pedagogico.pdf> (accessed August 27, 2018).

W3C, 2012. "Web Ontology Language (OWL)". W3C Semantic Web. Available at <https://www.w3.org/2001/sw/wiki/OWL> (accessed August 27, 2018).

W3C Web Ontology WG, 2004. "OWL Web Ontology Language Overview". W3C Recommendation. Available at <https://www.w3.org/2001/sw/wiki/OWL> (accessed August 27, 2018).

W3C OWL WG, 2012. "OWL 2 Web Ontology Language Document Overview (Second Edition)". W3C Recommendation. Available at <https://www.w3.org/TR/2012/REC-owl2-overview-20121211/> (accessed August 27, 2018).

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

Wohlin, C. et al, 2000. "Experimentation in software engineering: an introduction". Kluwer Academic Publishers, MA, USA.

Wynne, M; Hellesoy, A., 2012. "The Cucumber Book: Behaviour-Driven Development for Testers and Developers". Pragmatic Programmers LLC, Pragmatic Bookshelf, 309 pgs.

Zemmouchi-Ghomari, L. et al, 2013. "Process of Building Reference Ontology for Higher Education". Proceedings of World Congress on Engineering, Vol. III, 06 pgs.

Zubizarreta, J., 2009. "The Learning Portfolio: Reflective Practice for Improving Student Learning". Jossey-Bass, John Wiley & Sons Inc. San Francisco, 2nd edition, 354 pgs.

Appendix A

THE WRITTEN INFORMED CONSENT FORM

Termo de Consentimento Livre e Esclarecido

- (1) Você está sendo convidado a participar de um experimento relativo à pesquisa “ScrumOntoBDD: uma abordagem, baseada em Scrum, Ontologia e BDD, para o desenvolvimento ágil de software”;
- (2) Você foi selecionado por ser um profissional de Saúde, professor de Ensino Superior, especialista em Metodologias Ativas de Aprendizagem e por ter participado ativamente no desenvolvimento do projeto “Software de Gestão Pedagógica e Acadêmica para Cursos Baseados em Metodologias Ativas de Aprendizagem (SGPA-CBMAA)”;
- (3) O objetivo desse experimento é realizar uma análise comparativa da abordagem ScrumOntoBDD, proposta nessa pesquisa, com o método Scrum, utilizado no desenvolvimento do SGPA-CBMAA. Nesse experimento serão empregados o processo Pesquisa-Ação e uma entrevista semi-estruturada;
- (4) O experimento e a entrevista serão conduzidos pelo pesquisador Pedro Lopes de Souza, aluno de mestrado do Programa de Pós-Graduação em Ciência da Computação (PPG-CC) da Universidade Federal de São Carlos (UFSCar);
- (5) Sua participação nesse experimento consistirá em acompanhar a explanação do pesquisador das diversas fases envolvidas no desenvolvimento de uma aplicação, tanto utilizando o método Scrum quanto a abordagem ScrumOntoBDD, solicitar eventuais esclarecimentos durante essa explanação, responder às questões durante a entrevista e, se possível, tecer muitos comentários durante todo o experimento;
- (6) Esse experimento não oferece danos a sua saúde, mas eventuais constrangimentos ou desconfortos poderão surgir durante o uso de equipamentos computadorizados, tais como *desktop*, *notebook* e celular, ou devido ao tempo despendido na participação do experimento. Em relação à entrevista, as questões foram planejadas de forma a evitar possíveis constrangimentos ou desconfortos e caso ocorram, você poderá se recusar a responder ou mesmo interromper a sua participação a qualquer momento, sem qualquer prejuízo em sua relação com a instituição ou com o pesquisador;

(7) Para que se possa realizar uma boa análise posteriormente, todo o experimento será agravado desde que tenha o seu consentimento;

(8) Explicitação da liberdade do sujeito em recusar a participar ou retirar seu consentimento, em qualquer fase do experimento, sem penalização alguma e sem prejuízo ao seu cuidado:

(a) “A qualquer momento você pode desistir de participar e retirar seu consentimento”;

(b) “Sua recusa não trará nenhum prejuízo em sua relação com o pesquisador ou com a instituição”.

(9) Explicitação da garantia do sigilo que assegure a privacidade do sujeito quanto aos dados confidenciais envolvidos no experimento:

(a) “As informações obtidas através desse experimento serão confidenciais e asseguramos o sigilo sobre sua participação”;

(b) “Os dados não serão divulgados de forma a possibilitar sua identificação. Sua privacidade será garantida, pois você não será identificado como participante pelo seu nome, preservando o sigilo de sua identidade”.

(10) Não haverá qualquer ressarcimento relacionado a eventuais despesas decorrentes de sua participação no experimento.

Pedro Lopes de Souza

Programa de Pós-Graduação em Ciência da Computação (PPG-CC) - Universidade
Federal de São Carlos (UFSCar)

Caixa Postal 676, CEP 13.565-905, São Carlos-SP, Tel.: 16-33518233

Endereço e telefone do Pesquisador

Rua Dom Pedro II, 1133, Apto 42, Vila Monteiro, CEP 130560-320, São Carlos-SP,
Tel.: 16-33713820

Declaro que entendi os objetivos, riscos e benefícios de minha participação nesse experimento e concordo em participar.

São Carlos, 06 de agosto de 2018.

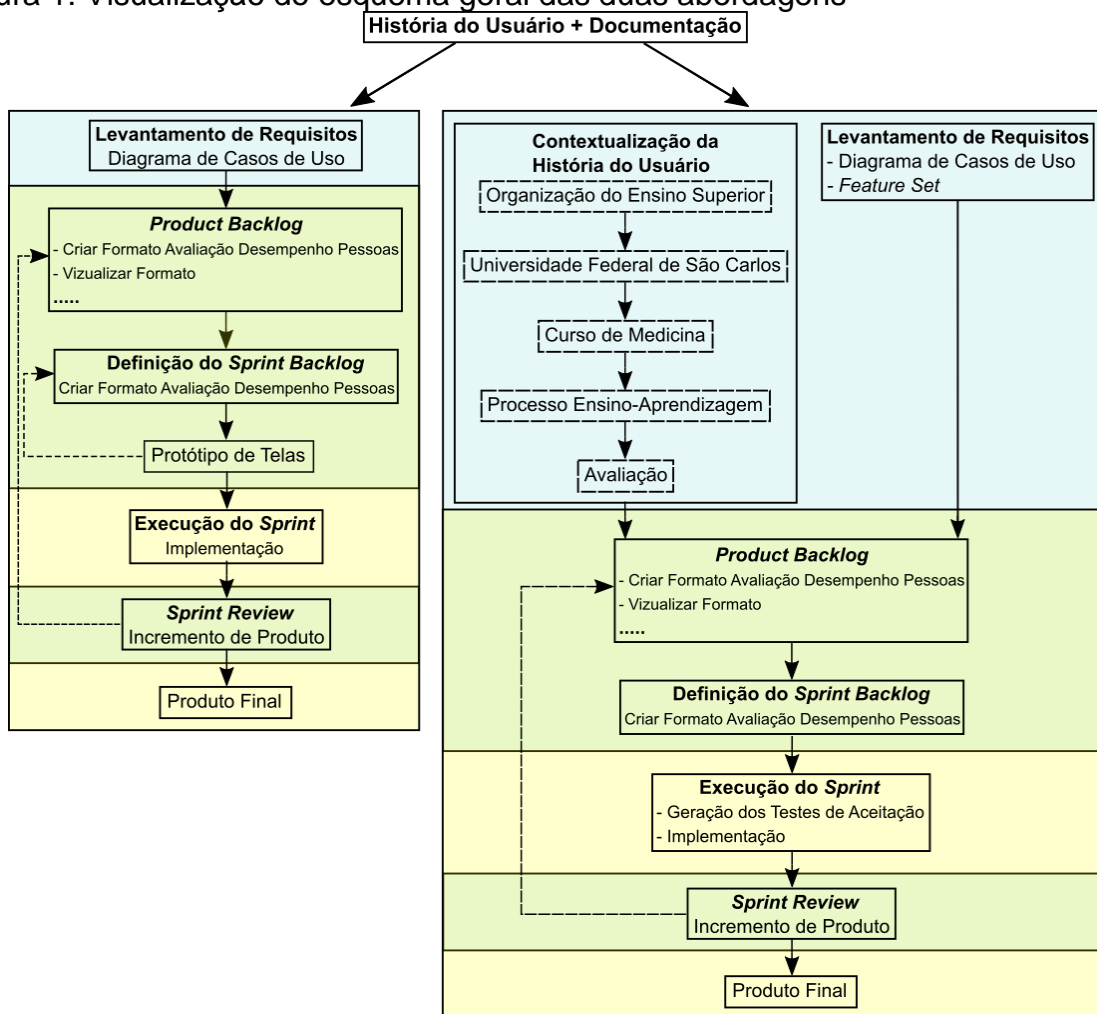
Assinatura do participante da pesquisa

Appendix B

THE INTERVIEWS ORIGINAL SCRIPT

ROTEIRO DE ENTREVISTA

Figura 1: Visualização do esquema geral das duas abordagens



- O esquema da esquerda representa a abordagem SCRUM pura, realizada pela TokenLab. O da direita representa a abordagem ScrumOntoBDD proposta no meu mestrado.
- Como você pode observar, ambas têm o mesmo ponto de partida: História do Usuário e a Documentação do Curso de Medicina.
- Relembrando a história do usuário que foi apresentada por você.

Figura 2: História do usuário em linguagem natural:

PO: Silva

Requisito: Criar uma avaliação de desempenho no processo ensino-aprendizagem do facilitador (*Formato na categoria Avaliação de Desempenho de Pessoas*)

História do Usuário:

Para que o estudante do curso possa realizar a **ADPEA do facilitador**, é necessário que o **coordenador do curso** (ou outro perfil que tenha essa permissão) crie o formato de avaliação ADPEA do facilitador. Para isso é necessário definir a **categoria desse formato de avaliação (avaliação de desempenho de pessoas)**, o curso, a oferta, a turma, a atividade curricular e a ação educacional.

Para cadastrar uma a avaliação ADPEA do facilitador, o sistema deve guardar os seguintes campos: o título desse formato; o **tipo de instrumento** (previamente cadastrado); um campo descritivo para registro de observações; o período a que se refere o formato contendo data de início e fim; o período que o formato ficará disponível para resposta pelos estudantes contendo data e hora de início e fim; e confirmar quais os grupos da turma que deverão responder a avaliação.

Após **definir o cabeçalho do formato**, o coordenador **define o corpo do formato**. **O corpo consiste em questões que podem ser do tipo dissertativa, ou objetiva de resposta única**. O corpo deve conter obrigatoriamente um campo denominado **conceito** que poderá aceitar as seguintes opções de resposta:

- Satisfatório;
- Precisa melhorar;
- Sem emissão de conceito.

O sistema deve permitir o reaproveitamento de outros corpos de formato previamente cadastrados, mas que possam ser editáveis. Além disso as questões desses corpos importados devem ficar na cor vermelha a fim de sinalizar que estas foram reaproveitadas. Ao finalizar a edição de um corpo de formato importado, a cor vermelha deve ser trocada para a cor padrão a fim de indicar que este foi validado com sucesso.

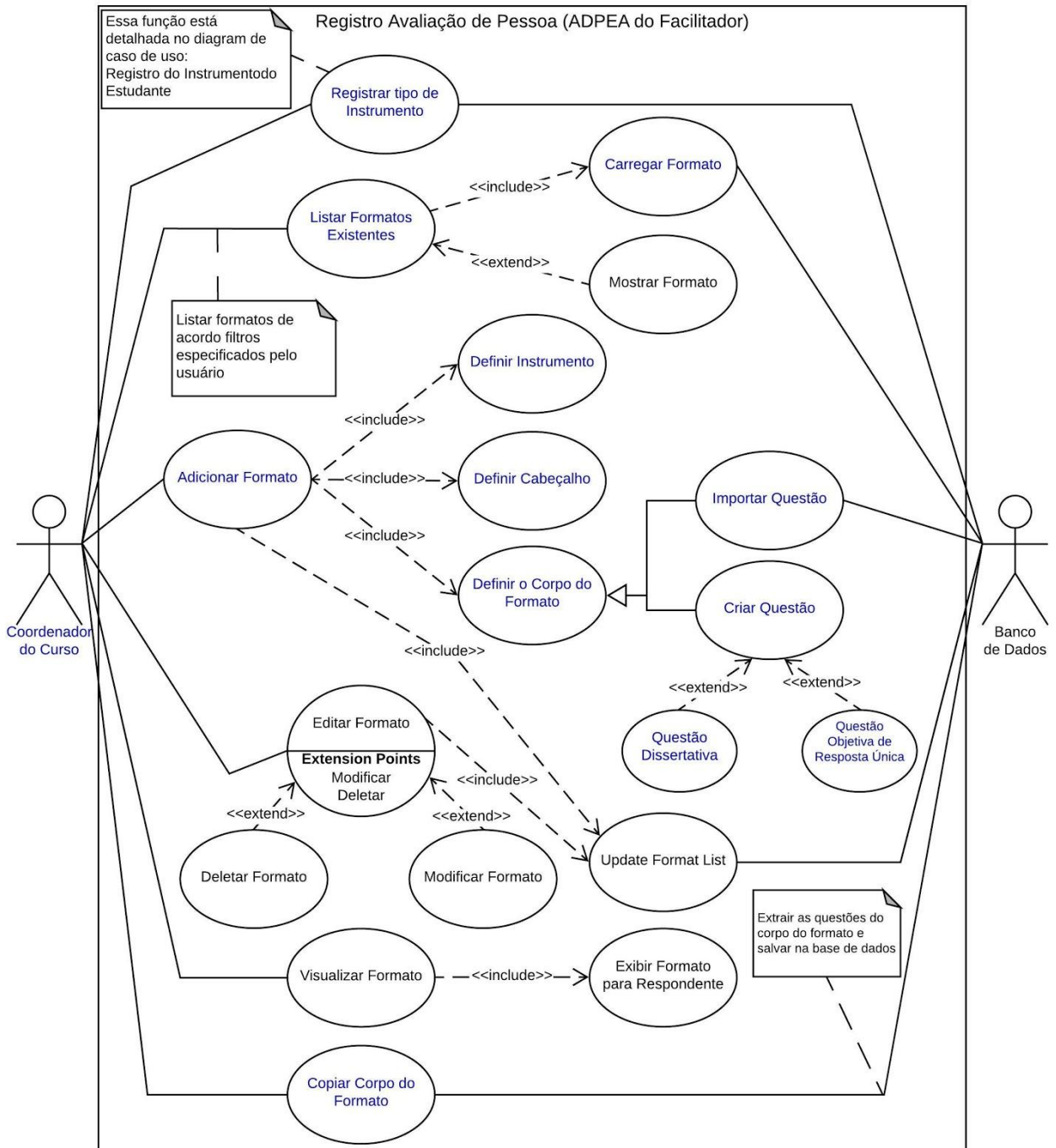
Exemplo de tipos de questões de um corpo de formato:

Dissertativa	<p>Como foi o desempenho do facilitador nas atividades de síntese provisória e nova síntese? Justifique.</p> <p>Como foi o cumprimento dos pactos de trabalho? Justifique.</p> <p>Comentários adicionais e/ou sugestões do estudante para o facilitador.</p>
Objetiva - Resposta única	<p>Como foi o desempenho do facilitador nas atividades de síntese provisória e nova síntese?</p> <p>a. Ótimo b. Bom c. Regular d. Péssimo</p> <p>Como foi o cumprimento dos pactos de trabalho?</p> <p>a. Ótimo b. Bom c. Regular d. Péssimo</p> <p>Outras questões específicas daquela atividade.</p> <p>a. Ótimo b. Bom c. Regular d. Péssimo</p>
Conceito	<ul style="list-style-type: none"> • Satisfatório • Precisa melhorar

- Em azul estão destacadas as características principais da história observadas pelos engenheiros de Software.
- Essas características é que vão ser reproduzidas na etapa seguinte: Diagrama de Caso de Uso.

Ciclo 1

Figura 3. Diagrama de Caso de Uso baseado na história do usuário:



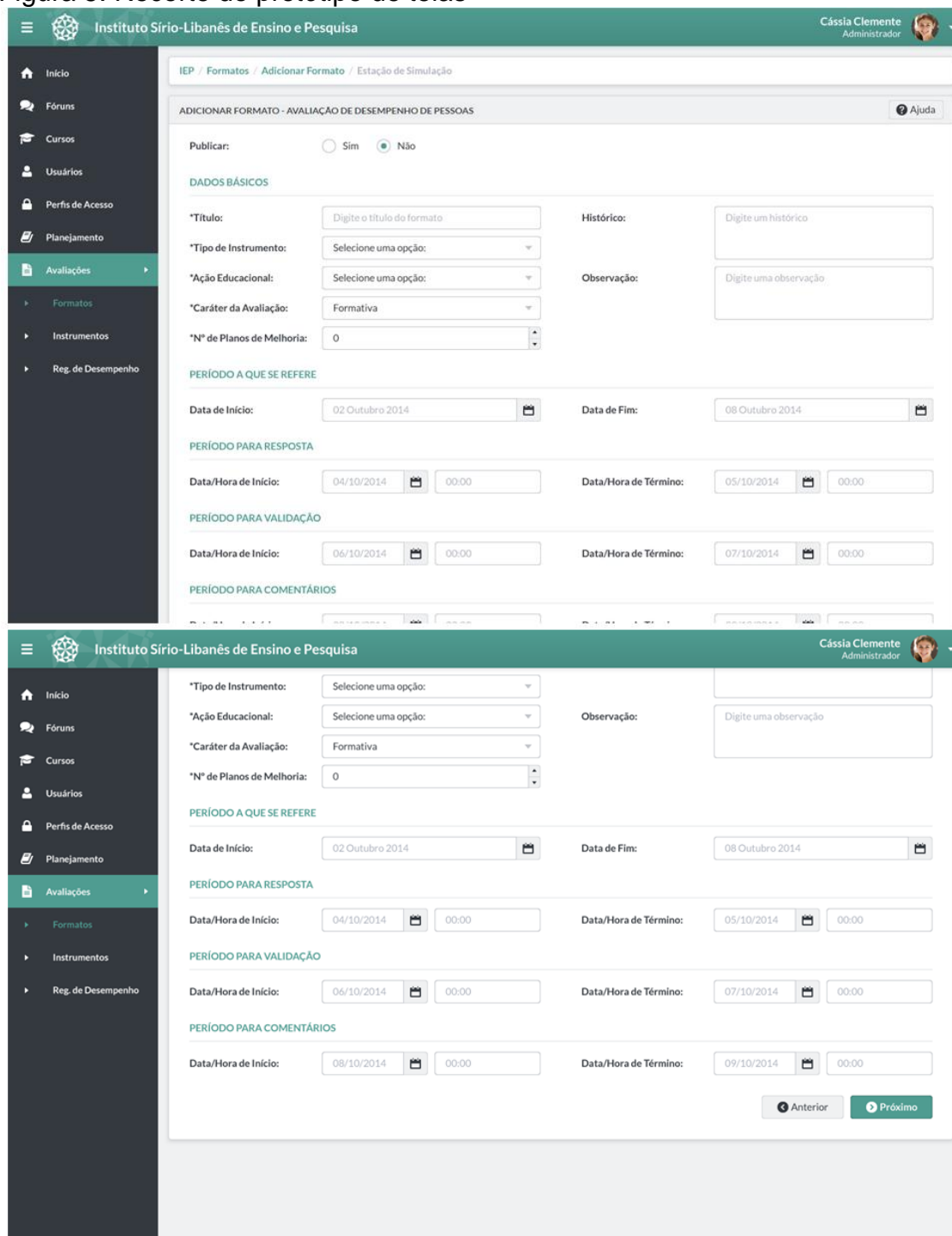
- Como você pode observar as características são resumidas nessa etapa.
- A partir dessas características resumidas neste diagrama, a equipe de desenvolvimento produziu o *product backlog*, uma lista, organizada por prioridades, de requisitos do sistema a serem desenvolvidos.

Figura 4: Product Backlog TokenLab

Project	Key	Summary	Issue Type	Status	Priority	Resolution	Assignee	Sprint
PEMAAP-Sirio-Libanes	SIR-2900	SIR-2874 [Front] Implementar replicação de formato	Sub-task	Resolved	Major	Done	Programador 1	Sprint 16, Sprint 17
PEMAAP-Sirio-Libanes	SIR-2899	SIR-2874 [Front] Implementar associação de grupos a formato de Avaliação de Desempenho de Pessoas	Sub-task	Closed	Major	Done	Programador 1	Sprint 16, Sprint 17
PEMAAP-Sirio-Libanes	SIR-2785	SIR-2080 [Front] Criação / edição de formato para permitir edição de alternativas	Sub-task	Closed	Major	Done	Programador 2	Sprint 16, Sprint 17
PEMAAP-Sirio-Libanes	SIR-2661	SIR-1849 [Front] Em formato, no cadastramento da questão, criar um tipo dissertativo e um tipo multipla escolha de resposta única	Sub-task	Closed	Major	Done	Programador 2	Sprint 16, Sprint 17
PEMAAP-Sirio-Libanes	SIR-2636	SIR-2601 [Front] Após salvar um formato, retornar para a tela de consulta de formato	Sub-task	Closed	Major	Done	Programador 1	Sprint 16
PEMAAP-Sirio-Libanes	SIR-2601	Como PO, gostaria que fossem sanados todos os bugs relacionados a Formatos	Bug	Closed	Major	Done	<i>Unassigned</i>	Sprint 17

- A partir do Product Backlog, foram selecionadas as primeiras atividades (as mais prioritárias) para compor o Sprint Backlog e criação dos protótipos de telas, como no recorte a seguir.

Figura 5: Recorte do protótipo de telas



- Você deve se lembrar dessas telas, e das reuniões em que foram feitas discussões e apresentadas críticas e sugestões.
- Em função dessas críticas e sugestões, foram atualizados os itens do Sprint Backlog e realizada a execução Sprint para implementação.
- Na próxima figura, você pode ver um recorte desta implementação final.

Figura 6: Código TokenLab

```

1 package br.com.tokenlab.pemaap
2
3 class Format {
4
5     String topic
6     Long bracket
7     String historical
8     Long assessmentBracket
9     Boolean mustBePublished
10    String notes
11    Long numberOfImprovementPlans
12    Date timeLengthInitialDate
13    Date timeLengthFinalDate
14    Date answerTimeInitialDate
15    Date answerTimeFinalDate
16    Date commentTimeInitialDate
17    Date commentTimeFinalDate
18    Date validationTimeInitialDate
19    Date validationTimeFinalDate
20
21    static hasOne = [appraisee: OfferOccupation,
22                    evaluator: OfferOccupation,
23                    course: Course,
24                    instrument: Instrument,
25                    levelFour: LevelFour,
26                    levelFive: LevelFive,
27                    intervieweeWithOfferOccupation: OfferOccupation]
28
29    SortedSet questionsStatements
30
31    static hasMany = [improvementPlans: ImprovementPlan,
32                    offers: Offer,
33                    questionsStatements: QuestionsStatement,
34                    respondedForms: RespondedForm,
35                    intervieweesWithAccessProfile: AccessProfile,
36                    schoolClasses: SchoolClass,
37                    workingGroups: WorkingGroup]
38
39    static belongsTo = WorkingGroup
40
41
42    static constraints = {
43        topic blank: false, nullable: false
44        bracket blank: false, nullable: false
45        historical blank: true, nullable: true
46        assessmentBracket blank: false, nullable: false
47        timeLengthInitialDate blank: true, nullable: true
48        timeLengthFinalDate blank: true, nullable: true
49        mustBePublished blank: false, nullable: false
50        notes blank: true, nullable: true
51        numberOfImprovementPlans blank: false, nullable: false
52        answerTimeInitialDate blank: false, nullable: false
53        answerTimeFinalDate blank: false, nullable: false
54        commentTimeInitialDate blank: false, nullable: false
55        commentTimeFinalDate blank: false, nullable: false
56        validationTimeInitialDate blank: false, nullable: false
57        validationTimeFinalDate blank: false, nullable: false
58
59        appraiser blank: true, nullable: true
60        evaluator blank: true, nullable: true
61        course blank: false, nullable: false
62        instrument blank: false, nullable: false
63        levelFour blank: true, nullable: true
64        levelFive blank: true, nullable: true
65        intervieweeWithOfferOccupation blank: true, nullable: true
66
67        improvementPlans blank: true, nullable: true
68        offers blank: false, nullable: false
69        questionsStatements blank: true, nullable: true
70        respondedForms blank: true, nullable: true
71        intervieweesWithAccessProfile blank: true, nullable: true
72        schoolClasses blank: false, nullable: false
73        workingGroups blank: true, nullable: true
74    }
75
76    static mapping = {
77        historical type: 'text'
78        notes type: 'text'
79    }
80
18 def retrieveFormatsForActivityEvaluator(formatQueryResultsTreeMap) {
19     JSONArray formatsForActivityEvaluatorByLevelAndCourseAndOfferAndSchoolClassJSONArray = new JSONArray()
20
21     formatQueryResultsTreeMap.values().each { formatQueryResult ->
22         JSONArray formatsJSONArray = new JSONArray()
23
24         formatQueryResult.formats.each { format ->
25             JSONObject formatJSONObject = new JSONObject()
26             formatJSONObject.put('formatId', format.id)
27             formatJSONObject.put('formatTopic', format.topic)
28             formatJSONObject.put('formatTimeLengthInitialDate', format.timeLengthInitialDate)
29             formatJSONObject.put('formatTimeLengthFinalDate', format.timeLengthFinalDate)
30             formatJSONObject.put('formatAnswerTimeInitialDate', format.answerTimeInitialDate)
31             formatJSONObject.put('formatAnswerTimeFinalDate', format.answerTimeFinalDate)
32             formatJSONObject.put('formatMustBePublished', format.mustBePublished)
33
34             formatsJSONArray.add(formatJSONObject)
35         }
36
37         JSONObject formatsJSONObject = new JSONObject()
38         formatsJSONObject.put('listTopic', FORMATS + formatQueryResult.topic)
39         formatsJSONObject.put('formats', formatsJSONArray)
40
41         formatsForActivityEvaluatorByLevelAndCourseAndOfferAndSchoolClassJSONArray.add(formatsJSONObject)
42     }
}

```

- O código gerou a implementação dos requisitos apresentados nas telas definitivas do programa, conforme recorte abaixo.

• **Figura 7: Tela criação do formato de Avaliação de Desempenho de Pessoas**

- Aqui se encerra o primeiro ciclo da Pesquisa-Ação. Realizar primeira rodada das perguntas semi-estruturadas
 - Qual foi sua participação no desenvolvimento dessa requisito específico do software?
 - E como você descreveria as etapas em que você participou mais ativamente?
 - Você apontaria etapas que foram mais críticas do ponto de vista da definição dos requisitos do sistema? (se sim, quais?)
 - Qual a sua opinião sobre o desenvolvimento do requisito como um todo?
 - Ou seja, considerando desde a sua primeira participação até a última tela
 - Como você vê o produto final como usuário do sistema?
 - O que, dessa experiência, você levaria para os próximos projetos de desenvolvimento de software?
 - Têm que ficar claras: as dificuldades que ela teve (principalmente de comunicação com os desenvolvedores) e sucessos
 - Têm que ficar claras: as dificuldades que ela teve com a interpretação dos diagramas apresentado até o momento
 - Caso ela não fale das dificuldades de comunicação, fazer uma pergunta mais direta: Sobre eventuais dificuldades de comunicação com os desenvolvedores, o que você poderia me dizer a respeito?

OBSERVAÇÕES:

- A. Sempre pedir exemplos ou esclarecimento de pontos relevantes
- B. Quando mencionar dificuldades, pedir mais detalhes

- Essa abordagem parte de um estudo inicial do domínio da aplicação, utilizando principalmente documentos desse domínio.
- No caso do desenvolvimento sistema, o domínio geral da aplicação é Educação Superior e o específico é o Curso de Medicina da UFSCar
- Esse estudo foi todo documentado através de ontologias
- Uma ontologia é uma especificação formal de um conjunto de conceitos, relações e outras distinções relevantes para modelar um domínio. Esse especificação define um vocabulário representacional, fornecendo o seu significado e as restrições para o seu uso coerente.

Ciclo 2

Figura 8: Organização do Ensino Superior

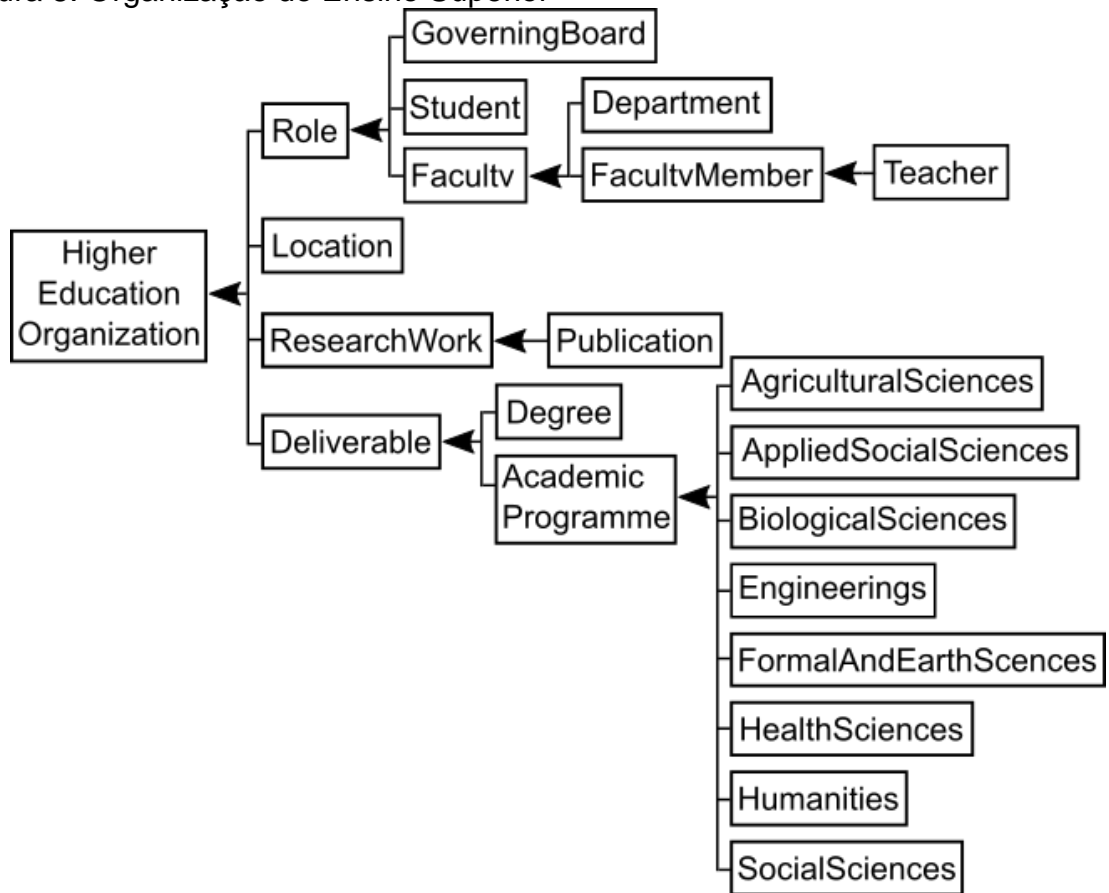


Figura 9: UFSCar e Curso de Medica

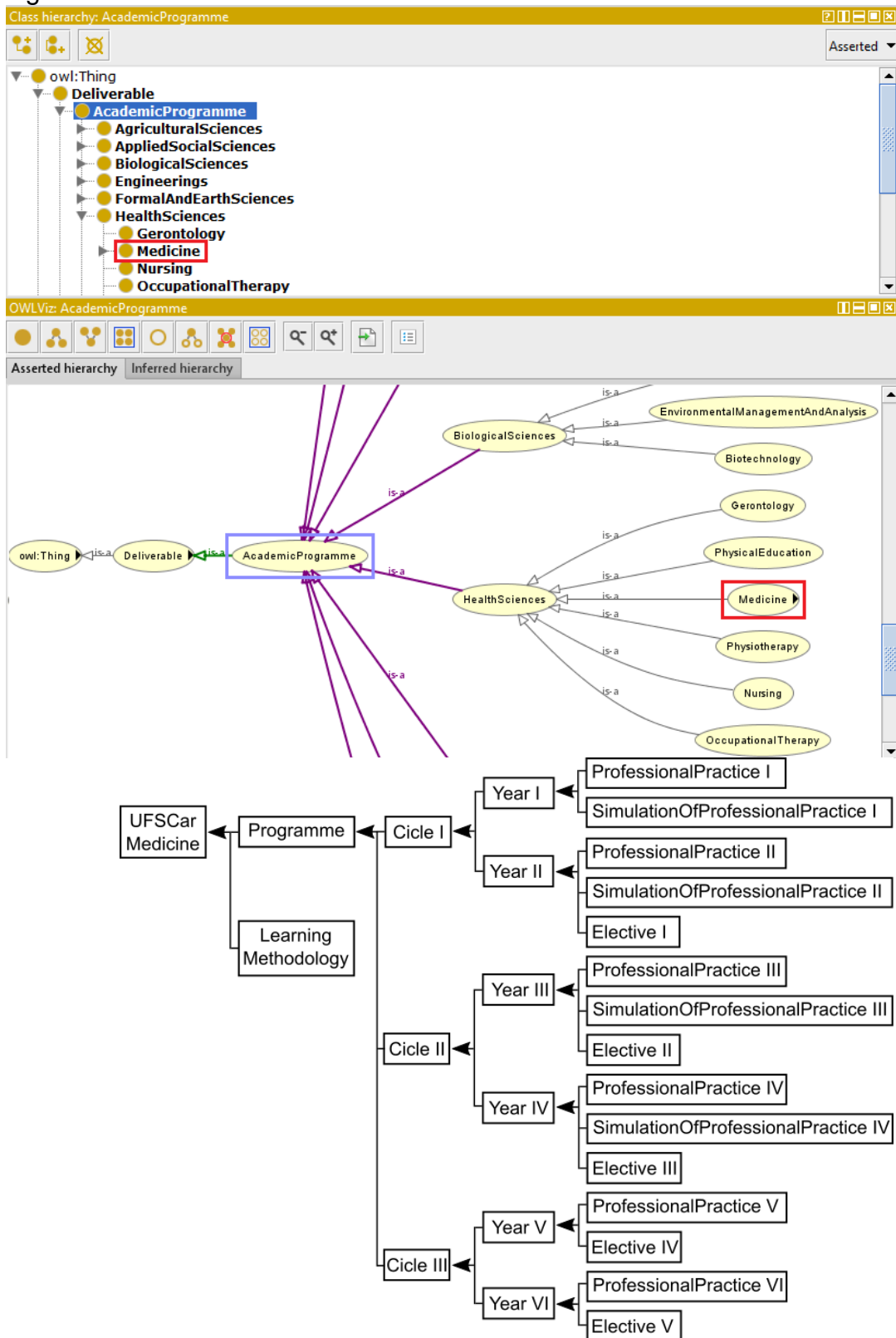


Figura 10: Processo Ensino-Aprendizagem

The image shows a screenshot of an ontology editor interface. It is divided into three main panels:

- Class hierarchy: Meeting:** A tree view showing the hierarchy of classes. The root is `owl:Thing`, which has several subclasses: `ConstructivistSpiralSteps`, `CurricularActivity`, `EducationalActivity`, `EvaluationInstrument`, `EvaluationProcess`, `LearningTrigger`, `Meeting` (highlighted in blue), and `Role`.
- Annotations: Meeting:** A panel showing annotations for the `Meeting` class. It includes an `Annotation` with the property `rdfs:comment` and the text: "The preceptor accompanies the students in the activities of health care in the services of the Network School and participates in the meetings to reflect on the practice of students, in partnership with the facilitators of UFSCar of the respective knowledge area".
- Description: Meeting:** A panel showing the description of the `Meeting` class. It includes an `Equivalent To` section and a `SubClass Of` section. The `SubClass Of` section lists three subclasses: `hasParticipationOf some Student`, `hasParticipationOf some Teacher`, and `itTriggers exactly 1 LearningTrigger`.

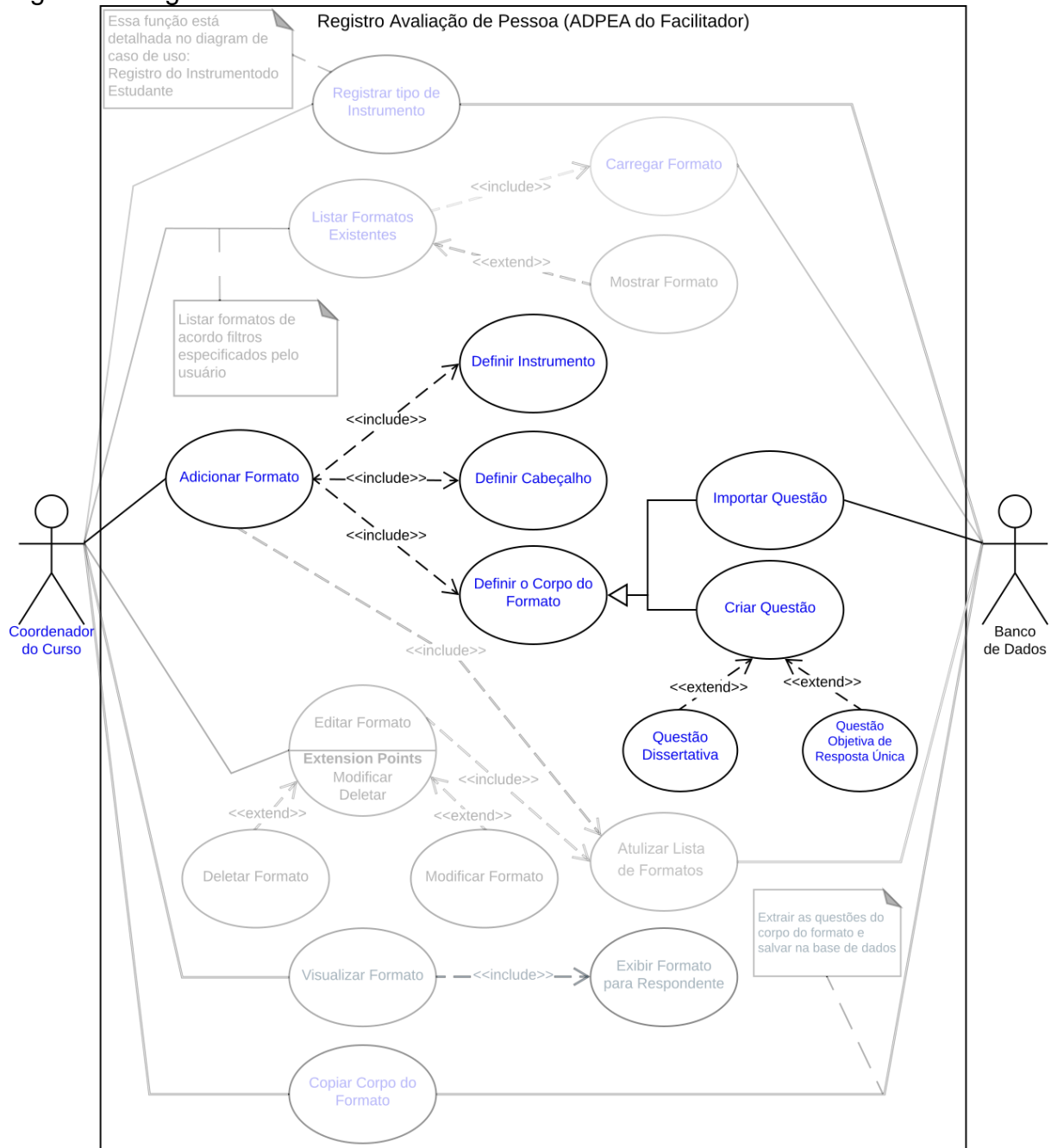
Figura 11: Avaliação de Desempenho de Pessoas

The image shows a screenshot of an ontology editor interface. It is divided into several panels:

- Class hierarchy: PeoplePerformaceEvaluationFormat:** A tree view showing the hierarchy of classes. The selected class is **PeoplePerformaceEvaluationFormat**, which is a subclass of **EvaluationFormat**. Other classes in the hierarchy include **ConstructivistSpiralSteps**, **CurricularActivity**, **EducationalActivity**, **CourseFormat**, **CurricularActivityFormat**, **EvaluationInstrument**, **EvaluationType**, **FormatBody**, **EssayQuestion**, **ObjectiveQuestion**, **FormatHeader**, **InteractionElement**, **LearningTrigger**, **Meeting**, and **Role**.
- Annotations: PeoplePerformaceEvaluationFormat:** A panel showing the **rdfs:comment** annotation for the selected class: "The People Performace Evaluation Format of curricular activities assess the student's ability to commit to their learning and training, contributing actively and co-responsible with the collective construction of knowledge, considering".
- Class description: PeoplePerformaceEvaluationFormat:** A panel showing the class description, including:
 - Equivalent To: (empty)
 - SubClass Of: **EvaluationFormat**, **hasBody exactly 1 FormatBody**, **hasHeader exactly 1 FormatHeader**.
 - General class axioms: (empty)
 - SubClass Of (Anonymous Ancestor): **hasType exactly 1 (Formative or Summative)**.
- Class hierarchy: FormatHeader:** A tree view showing the hierarchy of classes. The selected class is **FormatHeader**, which is a subclass of **FormatBody**. Other classes in the hierarchy include **ConstructivistSpiralSteps**, **CurricularActivity**, **EducationalActivity**, **EvaluationFormat**, **CourseFormat**, **CurricularActivityFormat**, **PeoplePerformaceEvaluationFormat**, **EvaluationInstrument**, **EvaluationType**, **FormatBody**, **EssayQuestion**, and **ObjectiveQuestion**.
- Annotations: FormatHeader:** A panel showing the **rdfs:comment** annotation for the selected class: "For registering a PATLP evaluation, the format must keep the following fields: the format title; the instrument type that was previously registered; a descriptive field for reporting observations; the period with start and end dates that the format will be available to be answered; and a confirmation of which actors should carry out the evaluation."
- Class description: FormatHeader:** A panel showing the class description, including:
 - Equivalent To: (empty)
 - SubClass Of: **(isSpecifiedBy exactly 1 EvaluationInstrument) and (isSpecifiedBy exactly 1 Checkbox) and (isSpecifiedBy exactly 3 TextField) and (isSpecifiedBy exactly 3 Calendar) and (isSpecifiedBy exactly 1 DropdownButton)**.
 - General class axioms: (empty)
 - SubClass Of (Anonymous Ancestor): (empty)
 - Instances: (empty)
 - Target for Key: (empty)
 - Disjoint With: (empty)
- Individuals by type: FormatHeader:** A panel showing a list of individuals of type **FormatHeader**:
 - DropdownButton (1)**: AssessmentCategory
 - Calendar (3)**: RelatedPeriod, CommentPeriod, AnswerPeriod
 - Checkbox (1)**: Group
 - TextField (3)**: NumberOfImprovementPlans, Observations, Title

- Com isso nós temos a terminologia principal do domínio da aplicação que será utilizada no desenvolvimento do sistema.
- Vamos levantar agora os requisitos
- Partindo novamente da História do Usuário + Documentação
- Retomando o Diagrama de Caso de Uso em que estão reproduzidas características principais da história observadas pelos engenheiros de Software, fiz um recorte do recurso (feature) para ilustrar as próximas etapas da nossa abordagem.

Figura 8: Diagrama de Caso de Uso com recorte do feature de interesse



Feature: Adicionar Formato de Avaliação de Desempenho de Pessoas do Facilitador

In order o estudante do curso possa realizar a ADPEA do facilitador

As a coordenador do curso

I want to adicionar o formato de Avaliação de Desempenho de Pessoas do Facilitador no sistema

Para facilitar o entendimento dessa feature, nós o refinamos em três grandes conjuntos de cenários que representam os principais comportamentos do usuário:

- Seleção do público alvo para o formato de avaliação a ser criado.
- A definição do cabeçalho do formato.

- A definição do corpo do formato.

Background 1 (conhecimento prévio):

Dado que os seguintes cursos e suas respectivas ofertas existem:

Curso	Oferta
Medicina	Oferta 2015 Oferta 2016
Programa de Pós Graduação em Gestão Clínica	Oferta PPGGC

Dado que as seguintes turmas existem:

Oferta	Turmas
Oferta 2015 Oferta 2016	Turma X (2015) Turma XI (2016)
Oferta PPGGC	VII - 2017

Dado que as seguintes atividades curriculares e suas ações educacionais existem:

Curso	Unidade Educacional	Atividade Curricular	Ações educacionais
Medicina	Necessidade de Saúde e Planos Terapêuticos I	Estações de Simulação I	- Oficinas de Trabalho - Simulações
		Situações-Problema I	- Oficina de Trabalho (Busca de Informações) - Processamento de Situações-Problema
	Prática Profissional I	Saúde da Família e Comunidade I	- Prática na USF - Reflexão da Prática
PPGGC	---	Linhas de Cuidado e Apoio Matricial	- ABP - Oficina de Trabalho - Plenário - Portifólio

Cenário 1.1: *O coordenador filtra o público alvo para o novo formato de ADP*

Given o coordenador seleciona Avaliação de Desempenho de Pessoas

And seleciona o curso de Medicina

And seleciona a Oferta de 2016

And seleciona a Turma XI (2016)

And seleciona a atividade curricular Estações de Simulação I

When clica na ação educacional Oficinas de Trabalho

Then o sistema verifica integridade dos dados

ScrumOntoBDD: an approach based on Scrum, Ontology and BDD for agile software development

And *carrega página para definição do cabeçalho do formato de avaliação*

Cenário 1.2 : Variações do cenário 1

Background 2 (conhecimento prévio):

Dado que as seguintes campos para dados básicos existem:

Dados	Relevância	Possíveis Instâncias
Título	Obrigatório	ADPEA do Facilitador
Tipo de Instrumento	Obrigatório	AD, ADPEA, AC, TP
Caráter de Avaliação	Obrigatório	Formativa, Somativa
Nro de Planos de Melhorias	Obrigatório	De 1 a 10
Observações	Opcional	Pode ficar em branco

Dado que, para a Turma XI (2016), os períodos a que se referem as avaliações são:

Atividade Curricular	Ações educacionais	Período
Estações de Simulação I	Oficinas de Trabalho Simulações	01/04/2016 a 10/04/2016 15/04/2016 a 25/04/2016
Situações-Problema I	Oficina de Trabalho (Busca de Informações) Processamento de Situações-Problema	01/05/2016 a 10/05/2016 15/05/2016 a 25/05/2016

Dado que, para a Oficinas de Trabalho, período para respostas e comentários são:

Período para Respostas	Período para Comentários
10/04/2016 às 00:00 até 14/04/2016 às 00:00	15/04/2016 às 00:01 até 20/04/2016 às 00:00

Dado que os grupos e seus participantes para o instrumento ADPEA são:

Turma XI (2016)	Respondentes/Avaliadores	Avaliado
Grupo A Grupo B Grupo C	Estudante	Facilitador, Preceptor
	Facilitador, Preceptor	Estudante

Cenário 2.1: *O coordenador define as variáveis do cabeçalho do formato ADP do facilitador*

Given *o coordenador define o título do formato como “ADPEA do Facilitador”*

And *seleciona o instrumento “ADPEA”*

And *seleciona caráter de avaliação “Somativa”*

And *define 1 para o nro de planos de melhorias*

And *define o período a que se refere “01/04/2016 a 10/04/2016”*

And define o período de respostas “10/04/2016 às 00:00 até 14/04/2016 às 00:00”

And define o período para comentários “15/04/2016 às 00:01 até 20/04/2016 às 00:00”

And seleciona “Grupo A” e “Grupo B”

And define respondentes e avaliadores como “Estudantes

And define avaliado como “Facilitador”

When clica em “Próximo”

Then o sistema verifica integridade dos dados

And carrega página para definição do cabeçalho do formato de avaliação

Cenário 2.1 : Variações do cenário 2.1

Background 3 (conhecimento prévio):

Dado que as seguintes formas de inserir questões no corpo do formato existem:

Alternativas	Tipo de questões
Criar	- Dissertativa - Objetiva de resposta única
Importar	- Dissertativa - Objetiva de resposta única

Cenário 3.1: Adicionar enunciado do conjunto de questões no corpo do formato de avaliação

Given o coordenador adiciona um novo enunciado de questão

When o coordenador clica em “Adicionar Enunciado”

Then o sistema abre uma nova tela com o campo texto para adicionar enunciado

Cenário 3.2: Salvar novo enunciado do conjunto de questões no corpo do formato de avaliação

Given o coordenador define o enunciado do conjunto de questões

And o coordenador define o título “ADPEA do Facilitador”

When clica “Salvar”

Then o sistema salva enunciado no corpo da questão

And e fecha a tela para adicionar enunciado

Cenário 3.2: Adicionar questões dentro do enunciado

Given o coordenador adiciona novas questões dentro do enunciado

When o coordenador clica “adicionar questões”

Then o sistema abre uma nova tela com os campos para adicionar questão

Figura 12: Modelo para especificação dos itens do Product Backlog

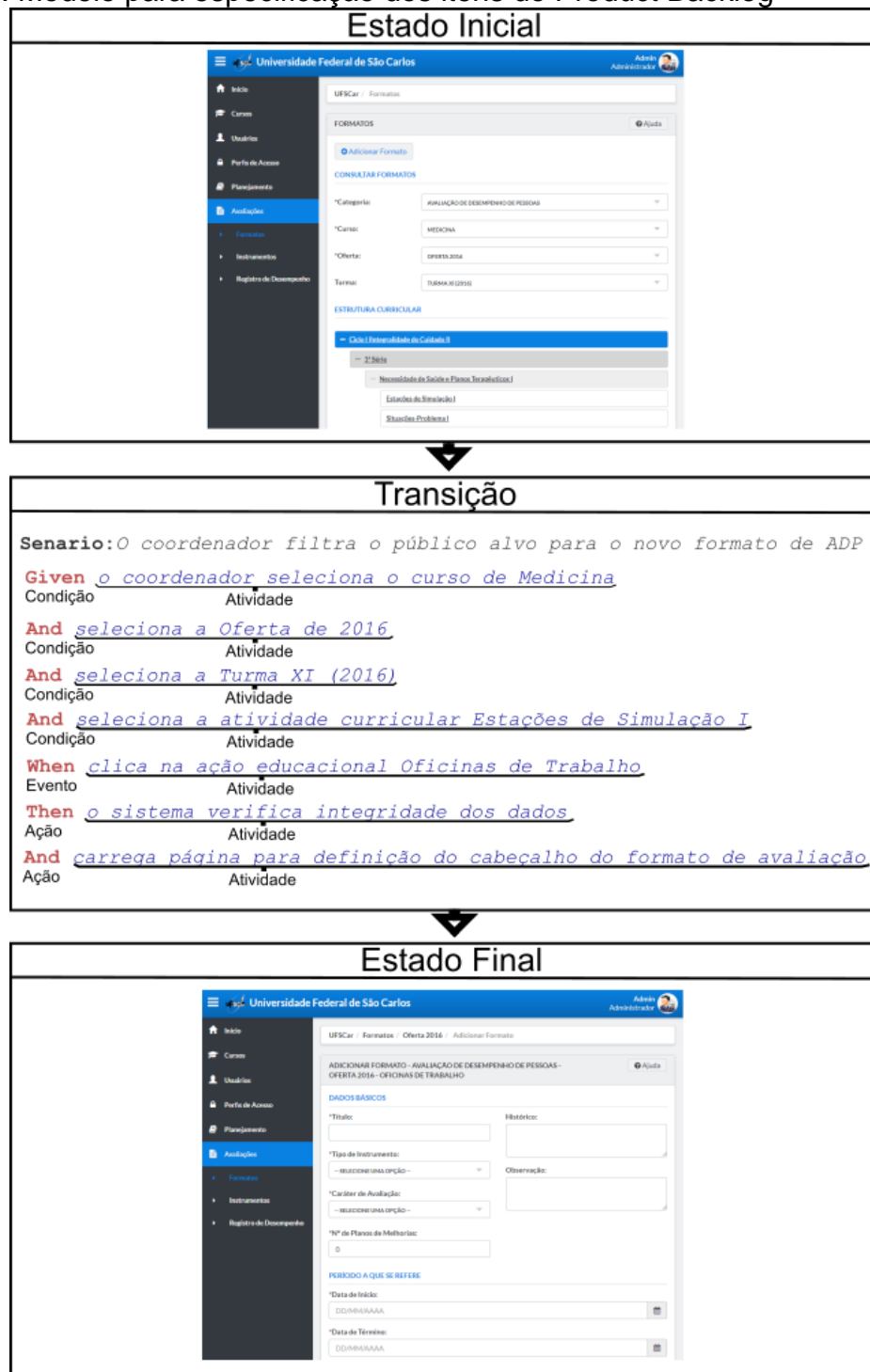


Figura 13: Ontologia do Sprint Backlog

The screenshot displays a Semantic Web browser interface with the following components:

- Class hierarchy: StateMachineElement**: A tree view showing the ontology structure. The root is `owl:Thing`, which includes `BDDTemplate`, `EvaluationFormat`, `EvaluationInstrument`, `FormatBody`, `FormatHeader`, `InteractionElement`, and `StateMachineElement`. `StateMachineElement` has subclasses `Action`, `Condition`, `Event`, `State`, and `Transition`.
- Individuals by type: PPEFSettings**: A list of individuals for the `PPEFSettings` class, including `Action (1)`, `Then`, `Condition (1)`, `Given`, `Event (1)`, `When`, and `Transition (1)`. The `PPEFSettings` individual is highlighted.
- Usage: PPEFSettings**: A section showing 15 uses of `PPEFSettings`, categorized into `PPEFSettings`, `Given`, `When`, and `Then`.
 - PPEFSettings**:
 - `PPEFSettings Type Transition`
 - `PPEFSettings isTriggeredBy Given`
 - `PPEFSettings isTriggeredBy When`
 - `PPEFSettings isTriggeredBy Then`
 - Given**:
 - `Given Type Condition`
 - `Given Category options "AVALIAÇÃO DE DESEMPENHO DE PESSOAS"^^element`
 - `Given Programme options "MEDICINA"^^element`
 - `Given Offer options "OFERTA 2016"^^element`
 - `Given CurricularActivity options "ESTAÇÃO DE SIMULAÇÃO I"^^element`
 - `Given Class options "TURMA XI (2016)"^^element`
 - When**:
 - `When Type Event`
 - `When clickOn EducationalAction actions "Oficinas de Trabalho"^^element`
 - Then**:
 - `Then Type Action`
 - `Then willBeChecked PPEFSettings`
 - `Then willBeLoaded PPEFHeader`

Figura 14: Trechos do Código Testes de Aceitação

```

public class PPEFSettingsSteps extends Steps{
... // setting variables and methods to support the scenarios
  @Given("the programme coordinator selects People Performance Evaluation")
  public void setCategory(DropdownButton Category){
    ... //set Evaluation format Type
    if (format.category.equals(PEOPLE_APPRAISAL)) {
      def currentWorkingGroupsIds = []
      format.workingGroups.each { workingGroup ->
        currentWorkingGroupsIds += workingGroup.id
      } def currentWorkingGroups = []
    }
    ... //set Evaluation format Type
  }
  @Given("selects the Medicine programme")
  public void setProgramme(DropdownButton Programme){
    ... //set Programme for this format
    def retrieveFormatsForCourseAppraisalByCourse(course) {
      def formats = Format.createCriteria().listDistinct {
        and { eq('category', COURSE_APPRAISAL)
          eq('course', course)
        }
      }
    }
    //set Programme for this format
  }
  @Given("selects the Offer 2016")
  public void setOffer(DropdownButton Offer){
    ... //set specific offer
    if (data.offer_id) {
      Offer offer = Offer.findById(data.offer_id as Long)
      format.addToOffers(offer)
    }
    ... //multiple specific
  }
  @Given("selects the Class XI (2016)")
  public void setClass(DropdownButton Class){
    ... //set class groups
    if (data.school_class_id) {
      SchoolClass schoolClass = SchoolClass.findById(data.school_class_id as
Long)
      format.addToSchoolClasses(schoolClass)
    }
    ... //set multiple class groups
  }
  @Given("selects the curricular activity Simulation Stations I")
  public void setCurricularActivity(Grid CurricularActivity){
    ... //set curricular activity
    if (curricular_activity.level_four_id) {
      levelFour = LevelFour.findById(curricular_activity.level_four_id as Long)
    } if (curricular_activity.level_five_id) {
      levelFive = LevelFive.findById(curricular_activity.level_five_id as Long)
      levelFour = levelFive.levelFour
    }
  }
  ... //set curricular activity
}
  @When("click on the educational action Workshops")
  public void setEducationalAction(Button EducationalAction) {
    ... //perform action checking errors
    if (jsonObject) {
      Format format = Format.findById(jsonObject.format_id as Long)
      if (!format) {
        messages += message(code: 'error.format.does.not.exist')
      }if (messages.size() != 0) {
        render(status: 422, text: messages as JSON)
        return
      }
    }
    ... //perform action
  }
}

```

```

@Then("the system verify the selected data integrity")
public void checkDataIntegrity() {
... //Check inserted data for black spaces or no Classes

    if (category.equals(PEOPLE_APPRAISAL)) {
        appraisee = OfferOccupation.findById(data.appraisee_id as Long)
        appraiser = OfferOccupation.findById(data.appraiser_id as Long)
        respondentWithOfferOccupation =
OfferOccupation.findById(data.respondent_id as Long)
        data.workingGroups.each { jsonWorkingGroup ->
            WorkingGroup workingGroup =
WorkingGroup.findById(jsonWorkingGroup.working_group_id as Long)
        if (workingGroup) {
            workingGroups += workingGroup
        }
    }
... //Check inserted data for black spaces or no Classes
}
@Then("loads the people performance evaluation format header page")
public void displayPPEFHeader() {
... //creating, persisting and display format header
    Format format = new Format(title: data.title,
        category: data.category as Long,
        historical: data.historical,
        assessmentCategory: data.assessmentCategory as Long,
        relatedPeriodInitialDate:
DateUtility.instance.stringToDate
(data.relatedPeriodInitialDate),
        relatedPeriodFinalDate:
DateUtility.instance.stringToDate
(data.relatedPeriodFinalDate),
        mustBePublished: data.mustBePublished,
        appraisee: appraisee,
        appraiser: appraiser,
        course: course,
        levelFour: levelFour,
        levelFive: levelFive,
        instrument: instrument,
        . . . // defining other fields
... //creating, persisting and display format header
} // next scenario
...}

```

Figura 15: Código Final

```

1 package br.com.ufscar.dc.ges.pemaap
2
3 class Format {
4
5     String title
6     String historicalNotes
7     Long category
8     Long assessmentCategory
9     Long numberOfImprovementPlans
10    Date relatedPeriodInitialDate
11    Date relatedPeriodFinalDate
12    Date answerPeriodInitialDate
13    Date answerPeriodFinalDate
14    Date commentPeriodInitialDate
15    Date commentPeriodFinalDate
16
17    SortedSet questionsStatements
18
19    static hasOne = [appraisee: OfferOccupation,
20                   appraiser: OfferOccupation,
21                   course: Course,
22                   instrument: Instrument,
23                   levelFour: LevelFour,
24                   levelFive: LevelFive,
25                   respondentWithOfferOccupation: OfferOccupation]
26
27    static hasMany = [improvementPlans: ImprovementPlan,
28                    offers: Offer,
29                    questionsStatements: QuestionsStatement,
30                    respondedForms: RespondedForm,
31                    respondentsWithAccessProfile: AccessProfile,
32                    schoolClasses: SchoolClass,
33                    workingGroups: WorkingGroup]
34
35    static belongsTo = WorkingGroup
36
37
38    static mapping = {
39        historicalNotes type: 'text'
40    }
41
42    static constraints = {
43        title blank: false, nullable: false
44        historicalNotes blank: true, nullable: true
45        category blank: false, nullable: false
46        assessmentCategory blank: false, nullable: false
47        numberOfImprovementPlans blank: false, nullable: false
48
49        relatedPeriodInitialDate blank: true, nullable: true
50        relatedPeriodFinalDate blank: true, nullable: true
51        answerPeriodInitialDate blank: false, nullable: false
52        answerPeriodFinalDate blank: false, nullable: false
53        commentPeriodInitialDate blank: false, nullable: false
54        commentPeriodFinalDate blank: false, nullable: false
55
56        appraisee blank: true, nullable: true
57        appraiser blank: true, nullable: true
58        course blank: false, nullable: false
59        instrument blank: false, nullable: false
60        levelFour blank: true, nullable: true
61        levelFive blank: true, nullable: true
62        respondentWithOfferOccupation blank: true, nullable: true
63
64        improvementPlans blank: true, nullable: true
65        offers blank: false, nullable: false
66        questionsStatements blank: true, nullable: true
67        respondedForms blank: true, nullable: true
68        respondentsWithAccessProfile blank: true, nullable: true
69        schoolClasses blank: false, nullable: false
70        workingGroups blank: true, nullable: true
71    }
72
73
74    def retrieveFormatsForActivityAppraisal(formatQueryResultsTreeMap) {
75        JSONArray formatsForActivityAppraisalByLevelAndCourseAndOfferAndSchoolClassJSONArray = new JSONArray()
76
77        formatQueryResultsTreeMap.values().each { formatQueryResult ->
78            JSONArray formatsJSONArray = new JSONArray()
79            JSONObject formatsJSONObject = new JSONObject()
80            formatsJSONObject.put('listTitle', FORMATS + formatQueryResult.title)
81            formatsJSONArray.put('formats', formatsJSONObject)
82            formatsForActivityAppraisalByLevelAndCourseAndOfferAndSchoolClassJSONArray.add(formatsJSONObject)
83
84            formatQueryResult.formats.each { format ->
85                JSONObject formatJSONObject = new JSONObject()
86                formatJSONObject.put('formatId', format.id)
87                formatJSONObject.put('formatTitle', format.title)
88                formatJSONObject.put('formatPeriodInitialDate', format.periodInitialDate)
89                formatJSONObject.put('formatPeriodFinalDate', format.periodFinalDate)
90                formatJSONObject.put('formatAnswerPeriodInitialDate', format.answerPeriodInitialDate)
91                formatJSONObject.put('formatAnswerPeriodFinalDate', format.answerPeriodFinalDate)
92                formatsJSONArray.add(formatJSONObject)
93            }
94        }
95    }
96
97 }

```

-
- Aqui se encerra o segundo ciclo da Pesquisa-Ação. Realizar segunda rodada das perguntas semi-estruturadas
 - Qual você acha que seria sua participação no desenvolvimento desse mesmo requisito nessa segunda abordagem, em comparação com a anterior?
 - Seria maior? Menor? Onde? Por que?
 - Você apontaria etapas mais críticas do ponto de vista da definição dos requisitos do sistema? (se sim, quais? Por que?)
 - Qual a sua opinião sobre o desenvolvimento do requisito como um todo usando essa segunda abordagem?
 - O que dessa abordagem você recomendaria (ou não) que fosse adotado em futuros projetos de desenvolvimento?
 - Têm que ficar claras: as dificuldades e facilidades que ela teve em entender as diversas etapas da abordagem

- Têm que ficar claras: as dificuldades e facilidades que ela teve em interpretar as figuras apresentadas nas diversas etapas da abordagem
- Caso ela não teça comentários a respeito da comunicação entre PO e desenvolvedores, fazer uma pergunta mais direta: Você acha que teria menos ou mais dificuldades de comunicação com os desenvolvedores utilizando essa abordagem? O que você pode me dizer a respeito?

OBSERVAÇÕES:

- C. Sempre pedir exemplos ou esclarecimento de pontos relevantes
- D. Quando mencionar dificuldades, pedir mais detalhes

Appendix C

THE AUDIO-TAPED INTERVIEWS TRANSCRIPTIONS

Entrevista 1: Silvia (PO)

Local: 06/08/2018

Data: São Carlos/SP, fora do contexto acadêmico ou profissional

Duração:

Áudio 1 - 0:43:30

Áudio 2 - 1:02:20

Convenções de transcrição: escrita convencional e sinalização de supressão de fragmentos inaudíveis, hesitações, repetições e falas paralelas através de (...) e acréscimos de explicitação de sentido através de []

Os nomes dos participantes e eventuais nomes citados foram substituídos por nomes fictícios a fim de manter os envolvidos em anonimato.

Ciclo I (áudio 1)

Observação sobre o fragmento do Backlog da empresa de software apresentado pelo entrevistador

12:02 a 13:04

Fico pensando que se eu tivesse lido isso aqui, o que eles fizeram, talvez eu pudesse ter ajudado mais (...) me veio essa ideia agora, porque, por exemplo, (...) tinha alguns problemas, por exemplo, eu não podia editar, tem alguns campos que não são editáveis depois que ele começa a valer. Então, são regras do sistema, que elas não ficaram transparentes pra mim. Então tem várias regras, pelo que eu tô vendo aqui (...) que na verdade eu não fiquei com a certeza de que regras que eles iam implementar e teve coisas que eles implementaram que não eram o que eu gostaria de ter, entendeu?

13:31 a 13:54

coisas que possam não ter ficado claras e na hora que eles passam pra cá [levantamento de requisitos] e na hora que eles vão fazer [product backlog], se eu não falei e eles têm dúvida, eles tomam uma decisão e fazem e depois que eu vou descobrir que não era bem aquilo (...) Ou fica sem transparência de qual é a regra que tá valendo. Então agora que eu estou testando, estou fazendo um trabalho de checar o que o sistema tá fazendo mas sem ter clareza daquilo que foi passado para eles fazerem mesmo.

14:00 a 14:13

Então, eu acho que nessa passagem aqui, eu acho que é um momento em que se perde, podem se perder coisas. Ou coisas que vão ficar com equívoco e depois vão precisar de acerto depois, né? De correção.

14:30 a 15:15

A metodologia é rápida, você consegue fazer as coisas caminharem mais rapidamente, mais tem esse outro lado: que você pode perder mesmo coisas. (...) Talvez se eu tivesse olhado isso, a tradução, eu poderia ter ajudado (...) a evitar trabalho desnecessário, esclarecer dúvidas e tal (...) Não que isso tenha tido grandes problemas. Mas no caso da avaliação em especial, que tem muita, tem regra, um processo cheio de detalhes, né? Acho que isso pode ter interferido mesmo.

Sobre as reuniões de protótipo de tela

16: 32 a 16:41

E por que que nao tem protótipo de tela nessa daqui [ScrumOntoBDD]? (...) foi uma opção?

17:54 a 18: 31

Primeiro vêm [as telas], a gente bate o martelo e depois eles implementam (...) e é um momento de repassar essa coisa das regras (...) pra chegar nessa coisa de cada formato, nós tivemos que conversar bastante.

18: 50 a 19:17

Isso foi falado, foi explicado qual era o negócio que a gente tava tratando, mas como isso é muito diferente, essa parte da avaliação, demorou um pouco pra eles, que é um aprendizado, né? Eu acho que teve um período aí que foi de aprendizado (...) entender essa avaliação, que é diferente (...)

19: 40 a 20: 18

Então, eu achei que foi demorado. Nao sei dizer demais, ou pouco, mas (...) a explicação que eu tenho é que eles tiveram que entender essa parte do negócio, né? (...) Porque é um processo inovador, que eles não conhecem. Quem teve uma formação tradicional de ensino não conhece esse tipo de avaliação, liga avaliação com prova.

21:42 a 21:50

Mas o primeiro desenho que eles trouxeram foi muito mais na linha de uma prova do que na de avaliação de desempenho do processo de ensino-aprendizagem.

Sobre avaliação do que faltou

22: 29 a 22:43

Aí não sei se é uma coisa de metodologia, né? Mas eu atribuo muito mais ao fato de ser um projeto diferente e as pessoas têm dificuldade de lidar com coisas que elas não estão acostumadas, né?

Sobre as telas

23:34 a 24:54

[As telas eram condizentes com o que se esperava?] Acho que mais próximas do que era, porque a gente teve como exemplo o IntegraMed (...) que foi a experiência prévia que tinha (...) desenvolvimento dessa parte de avaliação, então eu tinha esse outro sistema pra mostrar já uma coisa feita, né? (...) agora regra interna, eu acho que foi mais difícil deles aprenderem (...) e essa parte não ficou com o tempo necessário. Essa parte da avaliação. (...) O mesmo tempo que a gente dedicou a outras funcionalidades do sistema, a gente não fez a mesma coisa pra avaliação.

Sobre a classe formato

27:05 a 21:43

Instrumento e formato. Esses dois termos a gente acabou usando às vezes como sinônimo, e às vezes como coisa diferente. E aí a gente acabou definindo que isso aqui era na verdade tipo de avaliação, né? Se era desempenho de pessoas e era de avaliação de atividade ou avaliação cognitiva (...) avaliação tipo prova vamos dizer assim, tá?

34:42 a 35:25

Eu acho que quem tá demandando, no caso aí o P.O., né? (...) a pessoa, ou o conjunto de pessoas que conhecem o negócio, né? E acho que o desafio é conseguir conversar com quem vai fazer essa tradução pra linguagem do sistema. Esse diálogo, ele é importante, quer dizer, não dá pra achar que as pessoas lá sabem e quanto mais a gente consegue ter proximidade desses dois mundos, melhor vai ser o processo, melhor vai sair o produto.

35:38 a 36:31

A questão é quantas vezes você vai ter que ir e vir até ver que o negócio ficou do jeito que você queria (...) acho que essa metodologia, acho que ela tem como ponto positivo a coisa que é rápida, né? Vai fazendo e acredita-se que é em processo mesmo que esse conhecimento vai passando, né? Das pessoas de um lado e de outro nas suas respectivas funções e de alguma maneira um aprende com o outro. Isso vai ajudando a chegar lá no produto. Tem essa coisa da resposta rápida (...) Não demora tanto pra você chegar e ver uma tela, por exemplo, que isso dá uma concretude pra quem quer o sistema, né?

36:41 a 36:50

Por outro lado, se aparecer uma coisa que não ficou muito legal, o cara que fez o negócio, ele vai ter que (...) refazer, né?

Falando do IntegraMed

37:15 a 38:28

Eu fazia a mesma coisa: contava o que era, definia quais os campos que precisaria ter, se era um campo obrigatório ou não, acho que a gente gastou mais tempo e entrava mais em detalhes (...) acho que tinha mais tempo de contar, de fazer registro (...) eu lembro que a gente ia definindo, a gente tinha mais clareza se era um campo obrigatório ou não. Entendeu? (...) E na hora de ajustar o funcionamento, as regras, eu lembro que eu sentava às vezes do lado do moço que fazia, pra dizer o que que era, né? Olha, aqui é pra fazer assim, assim e assado (...) E às vezes tinha coisa que ele acertava na hora ali, do lado, na conversa comigo. Isso dava uma certa agilidade (...) mas era depois, né? Depois que a gente já tinha o produto quase que você tá validando, né?

Falando da experiência de Marília e da [empresa de software]

38:36 a 39:46

E a experiência anterior ainda que eu tive, que era lá em Marília, né? Na FAMEMA, eu tinha uma proximidade grande também com o pessoal que fazia desenvolvimento (...) e às vezes ajudava ali na conversa esclarecer às vezes até a ponto de ajudar a formular qual que seria a regra: se tal coisa e tal coisa, então faz isso, faz aquilo, faz aquilo outro (...) porque tenho um pouco de conhecimento dessa construção aí de coisas lógicas, é claro que banco de dados relacional vai complicando bastante, né? Mais complexo, mas às vezes eu gostava de olhar o (...) diagrama de entidade de relacionamento pra saber se eu fizer um pedido aqui assim, de onde é que ele vem buscar (...) às vezes eu entrava um pouco mais nessa questão técnica do sistema. E nesse processo que eu vivi aqui com an [empresa de software], talvez eu não tenha entrado muito nesse, nessa seara aqui, entendeu? Ficava mais por conta deles.

Avaliação geral

40:51 a 42:08

O que eu senti mais falta foi de ter um registro mais sistematizado das coisas. Não sei se é porque eu sou muito visual, eu preciso ter as coisas na mão e até pra traduzir os raciocínios, né? E acho que a gente ficava com a tela, nunca tinha visto a maneira como escreveu a estorinha, tinha uma validação em papel, assim. Ficava escrito. A gente ia mais acho que pelas telas mesmo (...) é que eu acho que pode ser que como eu conheço um pouco mais, talvez eu tivesse contribuído um pouco mais, talvez não era a minha parte do latifúndio, entendeu? (...) o PO não entra nessa parte. Por que não, se a pessoa tem essa possibilidade, por que não entrar também?

Ciclo II (áudio 2)

Quando o entrevistador explicou o que são ontologias

01:14 a 01:41

A gente precisa combinar, né? Instrumento significa isso, formato aquilo (...) metodologia de ensino ativa pra este projeto é isso, porque pra outro pode ser uma coisa, tem coisas que estão mais estabelecidas, outras menos (...) e acho que esse

processo [ontologias], então, ele pode ajudar o pessoal da área técnica entender mais do negócio.

06:55 an 07:44

Então ela [a ontologia] já vai dando a regra do negócio, aqui, né? De alguma maneira, ela vai contando coisas (...) então, na verdade, tudo que eu expliquei pra eles lá, como eles não leram o projeto, quer dizer, a gente até deixou, eles podem ter lido em algum momento, porque a gente vai traduzindo o que está escrito lá, né? (...) Mas a ideia de escrever tudo e tal, entendo que é pra você garantir que teve uma compreensão dessas (...) do que está lá no projeto, né? Do que define aquele processo, ou curso, etc e tal. Entendi.

09:09 a 09:20

Tem mais registros da explicação (...) tem uma sistematização das conversas, é isso.

Sobre a duplicidade dos técnicos

11:06 a 11:39

Os dois conversam comigo (...) ao mesmo tempo. Aí um vai desenhar umas coisas, outro vai desenhar outras (...) que coisa maluca que é isso! (...) [risos] porque se tem que dividir o trabalho de qualquer maneira, né? (...) na hora que um cara for fazer uma coisa e o outro for fazer outra e eles também não se conversarem, depois um não aproveitar o que o outro fez, não adiantou nada fazer tudo isso.

Sobre cenários

19:18 a 19:36

Agora já me deu um pressa, percebeu? (...) pra você ver a percepção que eu tenho (...) aquela outra parte [ontologias], achei que é uma coisa que vale à pena, que era uma coisa que eu sentia falta. Aqui eu já achei que já podia juntar o lé com cré, mas (...)

23:32 a 23:46

Talvez aqui esteja mais fácil pra eu ver o que o outro entendeu e se eu posso ajudar, melhorar ou uma coisa que eu não lembrei de falar lá atrás eu vejo aqui que está faltando, eu poderia falar nesse momento.

25:00 a 26:03

Só estranhei isso: que você trouxe (...) primeiro isso e depois. Eu acho que isso aqui tinha que vir depois. Só isso (...) tem que falar de enunciado, pra que que ele serve, né? Entendeu? Que enunciado é só um texto que a gente, inclusive, está procurando um nome melhor para enunciado (...) porque agora que a gente põe o sistema pras pessoas usarem, a gente vai percebendo que uma coisa que pra gente estava tão natural (...) então enunciado me remete a uma questão de prova mesmo (...) e não tem essa conotação aqui [no curso de Medicina] (...) Eu estou procurando um nome melhor pra isso daqui. (...) E qual que é a minha hipótese? Porque que ficou com esse nome? Lembra que eu falei que no começo quando eu falava de

avaliação, pra eles era prova. Não tinha outra possibilidade. E eles batizaram algumas coisas.

28:02 an 28:14

Tem alguns termos que eles são eu acho importantes, não dá pra fazer isso, né? Talvez fazendo assim a gente fique mais atento, não sei, tá?

Sobre o código BDD

31: 52 a 32:04

Ah! Entendi. Esse aqui é o (...) o BDD? (...) E os meninos lá não fazem isso.(...) E lá se eu quiser olhar eu não enxergo isso aqui.

Avaliação da solução pra demanda de um sistema flex

41:49 a 43:34

Na verdade isso que você fez, o sistema não tem essa coisa fixa, tá? Ele não funcionaria pra esse conjunto de coisas, entendeu? (...) agora se la no perfil de usuário tivesse valendo tipo de cliente, tudo bem (..) e hoje, olhando, eu voltaria, porque isso deu tanto trabalho pra um monte de coisa, que eu não faria mais. Eu deixaria: ou eu sou docente, ou eu sou discente, que é o nome mais genérico que gente tem (...) linguagem de escola, que é o que é mesmo, né? (...) Só que o jeito de dar a resposta pra isso [demanda de um sistema flex], eu acho, hoje, seguindo em frente, eu acho que não foi o melhor, entendeu?

44:49 a 45:20

Porque isso era nossa âncora pra separar um cara do outro (...) e ela não foi usada, seguiu em frente com esse negócio que não, o sistema nunca vai saber que nome que tem aquela criatura, o que que ela é, na verdade. E aí teve que criar depois lá dentro aquela outra coisinha lá (...) então foi dando um jeito lá pra frente mas talvez a gente não tenha parado pra olhar se não tinha uma coisa antes que...

46:15 a 47: 02

Poderia ter essa flexibilidade aqui na hora que a gente define o perfil dependente disso, né? Mas já foi (...) voltar e mexer nisso só se a gente tiver numa revisão muito grande do sistema pra reolhar isso, se não ele vai seguir como está mesmo (...) que ele está dando conta (...) agora ele não dá conta disso aqui que você propôs fazer. (...) você propôs uma coisa legal, porque tem especificidades da avaliação de desempenho que é o estudante avaliando o facilitador que são distintas do facilitador avaliando o estudante, e que eu agora vou ter de colocar na mão.

48:48 a 49:12

Agora dizer ah, seu eu tivesse ficado mais um mês, sei lá, essa hipótese, fazendo ontologia, não sei o quê, poderia ter evitado isso (...) essa coisa da flexibilidade (...) eu não sei, talvez, a gente tivesse arrumado outro caminho pra lidar com isso.

Avaliação geral

52:37 a 54:32

Olha, de uma maneira geral, o que eu percebi é que esta proposta, ela tem uma coisa que eu achava frágil na outra que era a questão dos registros, da possibilidade de ter o detalhe mais registrado, né? E tendo o registro, poder validar, talvez antes de fazer, de partir pra fase de (...) implementação, né? (...) então eu vejo vantagens. Essa coisa de combinar, então olha a gente vai usar o exemplo que eu dei aqui do enunciado, que ficou pra trás. Vou usar um termo e todo mundo sabe (...) então essa coisa, acho muito legal, vantagens em relação à outra (...) agora precisa também pensar na desvantagem (...) que eu consigo ver talvez fosse o tempo de desenvolvimento e o custo, porque vou ter que ter um cara que faz essa parte (...) e o quanto isso teria impacto no produto final (...) em termos de tempo e em termos de custo. Mas a minha impressão é que em termos de qualidade dos registros, da possibilidade de você até pensar melhor (...) eu vejo vantagens.

55:27 a 55:34

Não sei dizer se eu tivesse usando a outra metodologia, a gente ia encurtar esse tempo [tempo das reuniões sobre avaliação], entendeu?

56:05 a 56:37

E acho que a descontinuidade também acabou que ajudou a não fazer talvez o processo inteiro mesmo, né? Porque acho que a gente parou num momento em que eu teria que ter testado mais (...) e eu não fiz isso. Nem em 2015, parou, parou e quando eles entregaram em dezembro de 2017, naquele sufoco geral, eu demorei 3 meses pra conseguir fazer os testes, porque antes disso, eu não tinha condições de fazer.

Sobre ontologias

57:07 a 57:39

Eu acho que ela [ontologia] pode ajudar, então, na descrição mais detalhada das coisas, acho que essa questão da própria ideia da ontologia, né, que é registrar o que significa a, b e tal, e todo mundo tentar olhar, aproximar, né, o sentido da palavra que depois vai dar um sentido para aquela funcionalidade (...) no detalhe do negócio, eu acho que essa parte é que eu vejo como mais vantajosa mesmo (...)

1:00:12 a 1:02:06

[Ela desenhou uma figura dividida em 3 partes. Na esquerda os usuários, no meio o PO, Engenheiro de software e o engenheiro de ontologias, e na direita o time de desenvolvimento]

Vejo as vantagens e pra mim, agora, a principal desvantagem que eu vejo é essa coisa do tempo mesmo, do tempo pra eu ver alguma coisa concreta (...) apesar de que a proposta aqui não tem, né? os protótipos de tela, mas poderia, talvez se acrescentar isso aqui possa ajudar (...) porque pra quem não tem essa facilidade de pensar muito pra cá, de enxergar, ler esses montes de linhas, faz isso, faz aquilo, não é qualquer um que vai ter paciência (...) e mais uma coisa que eu pensei é que na hora que eu coloco mais uma figura de desenvolvimento [engenheiro de ontologias], eu vou aumentar o custo e tempo (...) mas é um aumento de custo e tempo que compensa porque eu vou ter um melhor resultado, menos retrabalho no processo? (...) pode ser. Mas isso eu não tenho como dizer pra você. A impressão

que eu tenho é que aumenta o custo e tempo, mas considerando as vantagens, pode ser que ele valha à pena, né?

Entrevista 2: Hilda (Engenheira de Software)

Local: 07/08/2018

Data: São Carlos/SP, fora do contexto acadêmico ou profissional

Duração: 1:45:14

Convenções de transcrição: escrita convencional e sinalização de supressão de fragmentos inaudíveis, hesitações, repetições e falas paralelas através de (...) e acréscimos de explicitação de sentido através de []

Os nomes dos participantes e eventuais nomes citados foram substituídos por nomes fictícios afim de manter os envolvidos em anonimato.

Ciclo 1

00:58 a 02:40

Eu não posso me considerar uma segunda P.O. porque eu não dominava o negócio. (...) eu me coloco mais do lado do desenvolvimento mesmo, mas eu me coloco do lado do desenvolvimento sem aquela responsabilidade de entender tudo com muitos detalhes pra desenvolver depois (...) então eu me coloco mais como auxiliar do desenvolvimento de software (...) se eu fosse da [empresa de software] eu me consideraria uma engenheira de software porque eu estaria imbuída da preocupação de desenvolver Então, eu estava mais com a preocupação de entender o que a P. O. falava e verificar se aquilo que o pessoal do desenvolvimento estava falando que ia fazer era o que eu, como desenvolvedora imagino que daria pra fazer (...) porque às vezes eles falavam assim: mas aqui não dá pra fazer. Eu falava: mas como que não dá? Dá pra fazer sim. Porque eu tenho a visão do desenvolvimento, eu sei que é possível fazer e às vezes eles falavam pra ela: não, isso aqui não vai dar. Então me explica porque, né? Então eu ficava nesse meio mesmo, entendendo o que ela estava falando e traduzindo se aquilo que eles estavam querendo dizer que iam fazer era alguma coisa que ia dar resultado e dando opiniões do que eu tenho de experiência de desenvolvimento, falando assim: por que que você não faz isso aqui? E aqui, por que que você não faz isso?

04:32 a 04:57

A utilização de uma metodologia ágil facilitou muito o desenvolvimento do sistema (...) o Scrum é ótimo pra você organizar funções dentro da equipe mas depois que você dividiu papéis, você tem que falar como fazer as atividades e isso o Scrum não faz

06:07 a 06:29

E eu acho que muita coisa fica na cabeça do Rivaldo (...) eu não vi como que ele exterioriza aquilo que ele conseguiu captar ali do que a Silvia passou pra ele (...) tanto que eles não disponibilizaram, você viu, né, naquele monte de material que tem (...) só o M.E.R.

06:42 a 07:00

Ele discutia várias vezes colocando o M.E.R. lá. Ele não discutia colocando os diagramas de caso de uso. Ele usou, pra poder fazer diversas discussões com ela uma ferramenta pra construção de interface que acho que você chegou a acompanhar alguma lá com ele.

07:17 a 07:36

Ele abria um documento igual um editor de texto, não word, um editor de texto tradicional e depois tem uns materiais (...) que ele construía: o protótipo navegável. Essa era a palavra que ele usava.

07: 56 a 08:22

Então, ele fazia no powerpoint, ele fazia a navegação da tela e voltava pra conversar com ela, mas pra validar aquilo que ela tinha falado no primeiro momento (...) então, ela contava, contava, contava, ele entendia, ele processava, ele escrevia num editor de texto tipo word pad

09:11 a 09.32

Então (...) recapitulando: o que que ele fazia? Ela falava, ele escrevia nesse documento de texto dele (...) depois disto ele montava o M.E.R. e esses protótipos navegáveis, e no próximo encontro trazia pra discutir com ela, pra validar se o que ela tinha falado, ele tinha entendido.

Sobre o BackLog

10:45 a 11:48

A Silvia, ela é uma usuária, que ela é diferenciada. Então, se você for fazer um desenvolvimento de software com uma outra pessoa que não tem o conhecimento que ela tem de passar a experiência pro desenvolvimento, seria mais difícil. Então, ela falava muito, ela conhecia o negócio e ela conhecia o processo de desenvolvimento de software, então ficava fácil dela encaixar aquelas duas coisas ali. Ela não conhece assim, de relações, um pra muitos, essas coisas todas ela não sabe porque ela não estudou, mas ela entende que você tem que transformar numa tabela, ela entende que acontece o milagre da multiplicação que talvez ele grave três ou quatro tabelas. Então, isso eu acho que foi um ponto positivo para que a [empresa de software] desenvolvesse isso de uma forma mais fácil e não sofresse tanto. Porque se fosse qualquer outra pessoa que tivesse que passar pra eles o que ela passou, acho que teria sido mais difícil o desenvolvimento,

12:43 a 13:03

Mostramos, então, o PRE, mostramos o IntegraMed, o caderno de curso, tanto o da UFSCar quanto o caderno de curso do Sírio, que diz lá tudo quanto é informação sobre o curso que você precisa ter, lá tem. E aquilo é um bom material pra você fazer essa análise de requisitos.

13:25 a 13:47

A parte de avaliação, por exemplo, ela era uma coisa muito complexa (...) até tem uma coisa da avaliação que ela não está completamente desenvolvida, né? Que é a segunda parte, que é onde você vai fazer os planos de melhoria.

15: 48 a 16:37

E essa avaliação do jeito que está feito, nós aplicamos desde (...) nós começamos com isso em 2014, acho que foi. Em 2015, terminou. 2016 ficou parado o ano inteiro (...) acho que no meio do ano de 2017 nós retomamos, né? Aí, nós tentamos colocar em 2016 pra rodar e nós não conseguimos professores que topassem lá na Medicina a usar. (...) Aí, então, quando que nós conseguimos usar a avaliação? Agora, no 1o semestre de 2018.

20:12 a 21:04

Então, voltando lá, né, que que a gente teve de dificuldade na época? É que o processo todo é muito grande. Então a gente tem que quebrar (...) Foi quebrado (...) talvez a falta de experiência dos desenvolvedores, onde eu me coloco, nessa nova forma de avaliar e como testar as partes (...) porque nós estamos acostumados com o, 1, 2, 3 ou A, B, C, D e E, né? (...) e chega no final, fecha: aprovado ou reprovado, né? E aí quando eles têm aquele Precisa melhorar, ele não está reprovado (...) ele ainda tem uma outra etapa pra refazer aquilo e aí quando que é esse momento que você bate o martelo e fala: aprovado ou reprovado?

23:47 a 23:56

E eles tentaram trabalhar essa forma, fazer entregas pequenas, só que essas entregas pequenas que eles faziam, a gente não conseguia colocar pra testar.

27:15 a 28:33

Mas o que eu me lembro é que o Ricardo, ele tinha essa visão do todo, não com tanto conhecimento quanto a Silvia tinha e algumas coisas que ele preparava no protótipo, elas não davam certo depois. Mas até aí, tudo bem, porque o protótipo serve pra isso mesmo (...) então quando eles optaram pela construção do protótipo, eu acho que foi uma escolha acertada. Eu só acho que o espaço das reuniões, eles eram muito demorados: até eles fazerem um protótipo, devolverem pra ela, e isso levava uma semana, né? Então, vinha numa semana, mostrava, eles trabalhavam o protótipo, vinha na semana seguinte. Isso que eu acho que tinha que ser um pouco mais ágil. Que foi o que eles tentaram fazer agora na segunda etapa. Eles mesmos perceberam isso: que tinha um intervalo muito gigantesco (...)

Sobre implementação

29: 51 a 30:15

Nessa segunda versão agora, existem muitas falhas, nessa outra parte que eles desenvolveram. E a Silvia pegou pesado. Porque na primeira parte, quem pegou pesado, fui eu. Eu fiz testes na primeira parte e eu me deparava com muitos bugs, alguns problemas bobos que, até eles fazerem essa correção e voltar, o processo era muito longo.

32:47 a 33:27

Então que discussões que deram, que eu lembro que deram discussões muito ferozes num primeiro momento? A interface. O Victor achava que ela não estava tão agradável quanto ele achava que deveria estar. Tem uma coisa que a gente questiona até hoje aqui da interface, né, que é isso aqui, ó (...) o aluno que entrar e ele tem um curso só, ele vai ter que fazer isso ó: um, dois, três, pra chegar no

ambiente. Então, isso foram coisas que deram muito questionamento, mas era mais com relação a usabilidade que com relação à funcionalidade propriamente dita

35:10 a 35: 51

O que eu me lembro é que o que era entendimento de conceito que trazia problema era muito pouco, [entrevistador]. (...) a não ser essa parte do plano de melhoria que ali não tinha jeito mesmo, né? (...) porque o plano de melhoria exige realmente (...) exige um conhecimento maior e aí precisava de tudo estar pronto pra conseguir trabalhar com plano de melhoria mesmo, se não, não tinha jeito, não.

37:10 an38:07

Até hoje a gente discute isso aqui ó: se eu entrar (...) aqui nesse portfólio individual, tem um problema de conceito que até hoje a gente briga com isso, tá? Então, esse portfólio individual aqui (...) tudo isso aqui são alunos que compartilharam com ela e aí, a professora, ela quer avaliar o portfólio. Porque, olha a palavra: portfólio individual. Esse portfólio é de quem? É da professora ou do aluno? Isso está gerando uma discussão até hoje. A partir do momento que o aluno compartilha com o professor, isso que ele compartilhou passa a compor o portfólio individual do professor e o professor está entrando nessa tela com o objetivo de ver o portfólio individual do aluno. Essas coisas não casam.

39:25 a 40:20

Uma ou outra coisa aconteceu de problema com relação a isso. Mas não foram tantas ainda porque o formato de avaliação tinha o IntegraMed, Aí a [empresa de software] conseguia ver o IntegraMed (...) se ela fosse usar só a fala dela[da P. O.], ela teria que trazer as avaliações por escrito e o processo por escrito, manual, do jeito que ela realiza. E aí, por que que ela não trazia? Porque ela tinha o IntegraMed pra mostrar. E ela sabia, no IntegraMed, o que estava bom e o que não estava. Então, ela tinha uma ferramenta que permitia ela mostrar o que eles usaram um dia e o que eles não gostaram naquela ferramenta, que aí a nova, que eles não queriam que fosse feita daquela forma.

41:16 a 41:28

Usamos o PRE, a primeira versão do PRE, usamos o IntegraMed e usamos os cadernos de curso e a Silvia. Tudo isso foi fonte pra coleta dos requisitos

41:55 a 42:17

Mesmo ela entendendo todo esse processo, ela sempre tentou detalhar tudo, mas às vezes alguma coisinha num vinha e aí essa coisinha só vinha quando via o protótipo, ou (...) aí que ela via que a gente não estava entendendo, ela falava: vamos abrir o IntegraMed, aí abria o IntegraMed e discutia com eles.

Análise do método Scrum

44:19 a 45:07

O que eu levaria? O protótipo eu levaria, porque eu vi que foi uma coisa que deu resultado (...) porque era um meio palpável de se discutir (...) porque senão fica tudo muito na cabeça, tudo muito nos diagramas. Então, o protótipo mesmo ele sendo navegável, ele deu pra dar uma ideia (...) o que eu não levaria (...) pra mim, a

[empresa de software] tinha que ter levado em consideração um pouco mais, eu estava lá, a experiência de algumas coisas que eu apontava pra eles e eles não queriam aceitar naquele momento.

46:13 a 46:38

Mas só que eu acho assim, ó: a gente fez uma reunião na segunda-feira, e a minha equipe, agora eu vou passar pra minha equipe, minha equipe vai desenvolver. Então, terça, quarta, quinta e sexta, a equipe cheia de dúvida, ela não consegue produzir. Então, nesse momento aqui, tinha que ter um contato maior com o P.O. No momento da coleta de requisitos. E eu acho que isso daqui não aconteceu muito, tá?

48:14 a 50:39

Então eu acho que essa metodologia, ela é muito pesada com relação ao desenvolvimento, né? Ela exige bastante, e com isso a produtividade cai. (...) essa produtividade, eu acho que ela demorou muito (...) uma outra coisa que eu acho que essa outra ferramenta pra construir os protótipos que eles usaram, aí depois você tinha que programar tudo na outra linguagem. Então, talvez essa construção do protótipo, ela ser numa ferramenta que já vai ser a sua ferramenta depois (...) e eu cheguei à conclusão [a partir da experiência com uma do Oracle] que ela é fantástica pra protótipo, porque eu desenvolvo muito fácil com ela e com a possibilidade do usuário colocar dados nela. Então, eu acho que se eu desenvolvo, e desenvolvo muito rápido mesmo, e dou pro usuário brincar e o usuário gosta, aí paralelo a isso, já que é pra usar uma diferente mesmo, porque eles usaram uma diferente, e a que eles usaram, o usuário não conseguia brincar nela, pra botar dados e pra sentir.

50: 58 a 51:22

Então, eu ganhei [na produtividade], eu testei, o usuário brincou, ele viu que ia dar certo, não ia, e o outro fazendo aqui do lado já na linguagem definitiva. Então, isso eu acho que é um ponto negativo, porque a produtividade ali, pra mim, nessas tecnologias que eles usaram, ela é baixa. Mas eu não sei, [entrevistador], se existe ferramenta que vai dar um salto ali na produtividade, não sei.

Ciclo 2

Sobre ontologias e BDD

1:05:18 a 1:08:13

Primeiro, interessantíssimo no sentido de: tem um padrão pra se seguir. Porque na forma em que os requisitos foram coletados. Então, quando eu dou aula e eu tenho que falar pros alunos que eles têm que entender o requisito, eu tenho uma dificuldade muito grande em fazer num primeiro momento os alunos entenderem o que são requisitos, né? (...) o que é pior, que vem depois: como que, a partir de um requisito, ele identifica os casos de uso e como, a partir de um requisito, ele identifica as classes. Quando esse requisito está escrito na forma de um texto. (...) como que daquele texto eu consigo identificar quais são as ações? (...) então, na minha cabeça, como eu trabalho há muitos anos com desenvolvimento, na hora que eu leio um texto, o meu mapeamento, de um texto enxergar casos de uso e de um texto enxergar classe, ele é uma coisa muito automática. (...) Eu consigo, mas só

que eu tenho 30 anos de desenvolvimento. Se você fizesse essa pergunta pra mim, lá quando eu saí da universidade, em 85, eu não ia conseguir ter essa agilidade que eu tenho hoje. (...) e aí eu imagino a [empresa de software] com um monte de pessoas trabalhando pra ela, e pessoas novas, que não têm tantas experiências assim (...) aí, quando eu vejo um modelo como esse, então, que você tem lá a feature, a an, I want to (...) [lendo estruturas em inglês do BackLog] eu acho que isso é um modelo pra você seguir, pra você fazer requisito. E aí você insere o usuário esse contexto e as coisas já vão sair mais ou menos do jeito que elas têm que ser depois. (...) Eu vejo vantagem nisso daqui. É o fato de você ter um padrão e você usar esse padrão pra fazer requisito.

1:08:21 a 1:09:39

[na empresa de software] eu acho que existiu muito a descrição de um texto (...) é o que a gente tem, né? É o que todo mundo usa. Mas acho que precisamos mudar.

1:11:20 a 1:11:43

E quando você escreve dessa forma [redação de um texto], ela te dá possibilidade de dupla interpretação. Então, quando eu uso um modelo desse [ontologias], ele está mais com o formalismo definido, daí quando eu aplico esse formalismo, ele dá menos possibilidade de dupla informação (...) ambiguidade. Então eu consigo extrair o requisito de uma forma mais fácil.

1:15:58 a 1:16:32

Mas por outro lado, qual que é a crítica que tem com relação aos métodos formais? Eles são mais difíceis de aprender. Porque hoje todo mundo sabe escrever, todo mundo sabe (...) abrir o editor de texto e começar a escrever. Tudo bem, né, que no primeiro momento ele não vai conseguir interpretar, mas eles acham que aquilo é melhor do que fazer isso. Então, existem várias críticas nesse sentido. Eu, depois de todos os anos meus de experiência, eu sei que isso daqui é fantástico, que se eu conseguir ensinar isso pras pessoas, eu consigo coletar requisito mais rapidamente e consigo escrever sistema com menos erros.

1:16:43 a 1:17:33

Ai, eu vou te falar outra crítica que eu faço às empresas que começam a adotar o Scrum. Scrum é lindo (...) qual foi o problema, [entrevistador]? [da experiência de uma firma em Araçatuba que introduziu o Scrum há mais de 10 anos] Eu acho assim: o Scrum, ele é ótimo pra você organizar funções dentro da equipe (...) agora, depois que você dividiu papéis, você tem que ter isso aqui, ó, pra falar pra ele como é que ele faz a atividade dele.

1:21:56 a 1:22:12

Então, eu acho que isso, ele complementa o Scrum sim. E precisa. Porque a experiência que eu tive, não existia essa formalização [ontologias e BDD] de como é que você vai complementar as outras atividades que você tem que fazer.

Sobre a criação de uma terceira figura (engenheiro de ontologias)

1:23:35 a 1:24:05

Quando você vai desenvolver um software, você tem que entender como as coisas funcionam. Você tem que entender o processo. Porque que se você não entender como o processo funciona, e você for implementar do jeito que você quer, aquilo que você vai implementar talvez não vá caber naquele processo. Porque o processo está de um jeito e você vai implementar de outro (...) Ou, se você for implementar exatamente do jeito que o processo está, talvez você não vá ter grande sucesso, porque talvez aquele processo precisaria ser modificado para depois você vir implementar.

1:27:02 a 1:28:01

Então, pra mim, não tem diferença nenhuma, só que eu vivi tudo isso (...) eu não informatizava nada antes do cara do AIM (????) entrar em ação. Mas eu trabalhei numa empresa que me proporcionou isso. Então essa empresa tinha esse cara do AIM (...) o dia que esse cara saiu, eu comecei, então eu chego nos lugares, eu falo assim: mas o que você quer fazer? (...) essa é a diferença minha pro pessoal da [empresa de software], que na hora que pediu pra colocar os dados, colocou, na hora que pediu pra sair o dado, não estava onde tinha que estar. É a experiência da vida que faz a gente ser assim.

1:28:40 a 1:29:05

Eu, enquanto desenvolvedora, tenho necessidade dessa figura, né. Aí, tudo bem, é alguém que foi treinado pra isso. Do mesmo jeito que na computação a gente já tem tantas especializações, tem o cara que faz a interface, tem o que cuida do banco de dados, tem o que cuida da rede, né? Então, tem aquele que depois vai ter que trabalhar em conjunto com o engenheiro de software pra poder construir a ontologia. Não vejo problema nenhum.

1:29:37 a 1:32:04

Aqui, então, ela [a ontologia] está muito mais fácil do usuário entender do que aqui, né? Porque aqui você instanciou praticamente, então aqui você deu valores pra ela. Aí ele enxerga. Eu não sei se todos os usuários - porque a gente está analisando Silvia. Silvia é um ser que não é comum como sendo um usuário - (...) como P. O. Os P.O. não são do jeito que a Silvia é (...) eles vão falar muito rapidamente o que eles querem (...) Então, a gente não pode levar em consideração Silvia. Silvia vai entender isso aqui perfeitamente, tá? Porque Silvia tem um pezinho na modelagem. A Silvia, ela consegue ver o que é um banco de dados, o que são as tabelas, como as coisas se relacionam. Eu não sei se essa ontologia seria bem aceita por alguns desses outros P. O. que passaram na minha vida. Mas se ele tiver com um pouco de disposição pra poder entender, isso é muito mais claro.

1:32:08 a 1:33:49

Só que eu tenho certeza que isso daqui pra equipe de desenvolvimento, que é onde não pode gerar dúvida, isso daqui é ótimo (...) então, essa conversa [entre engenheiro de software e desenvolvedor] ela tem que ter uma forma, ela tem que ter um formato, não pode ser só verbal. Porque se ela for só verbal, eu falo pra você fazer uma coisa, você vai fazer o que você entendeu. Será que você entendeu o que realmente eu queria dizer? Aí, aqui, eu vejo que isso tem grande utilidade. Então, entre nós, eu sei que isso tem grande utilidade.

1:35:03 a 1:36:13

Isso aqui é sucesso com a Silvia. Eu não garanto que isso vai ter sucesso com aquele outro usuário que quer te falar rapidamente e vazar da reunião porque ele tem um monte de outras coisas pra fazer.. Porque muitas vezes o P. O. também tem isso, né? Ele tem que sair do serviço dele pra vir te atender, pra passar alguns requisitos e depois voltar pro serviço. E às vezes tá lá lotado de coisas pra fazer (...) ele faz de qualquer jeito e às vezes você tem que arrancar (...) por isso que às vezes uma semana é muito tempo. Você vai pra casa, pensa, e fala: mas aqui está faltando alguma coisa, eu acho que ele não falou pra mim alguma coisa (...) então, pra eu construir isso depois que eu conversei com o P. O., perfeito. Se todos os POs vão entender isso, tenho dúvidas. Aqueles que são do tipo da Silvia, sim. Com certeza. E essa conversa com os desenvolvedores? Ótimo, porque o desenvolvedor, ele vai entender isso daqui. Ele é técnico (...) Isso daqui ele vai entender sim. Então essa é a visão que eu tenho (...) que eu trago comigo ai dos meus 30 anos de experiência.

Análise da abordagem ScrumOntoBDD

1:36:58 a 1:37:48

É aquilo que eu falei pra você, (...) porque o formalismo, num primeiro momento, todo mundo acha que o formalismo toma tempo. Mas ela [a Silvia] ela está vendo o tempo que ela está levando agora pra fazer todos os testes, que as coisas foram feitas com bugzinhos que talvez seriam solucionados, porque bug não é falta de compreensão, é inexperiência mesmo, e talvez seriam solucionados aqui, se nesse momento a gente tivesse dito um pouco mais (...) se você tem um formalismo, aí naquele formalismo ele já está falando pra você o que que a tela tem que ter, o que ela tem que apresentar, o que ela não pode ter (...) [entrevistador], nós estamos testando desde janeiro, nós já estamos em agosto! (...) é muito tempo e ainda está lotado de problema.

1:39:16 a 1:39:28

Ele entregou em janeiro. Tinha que estar tudo testado. Tudo isso tinha que estar funcionando. Nossa Senhora! Você não imagina o tanto de versão que já teve que vir, o tanto de (...) quanto de backend

1:40:31 a 1:40:48

Então, depois que você aprendeu, isso daqui você tira de letra e na hora que você entregar, realmente você entregou. Erro (...) sempre vai ter (...) mas não pode ser do tanto que está aqui.

1:41:54 a 1:42:00

Então, consome um pouco mais de tempo? Consome sim. Só que eu garanto pra você que você vai ter menos erro.