Improving Agile Software Development with Domain Ontologies

37

Pedro Lopes de Souza, Antonio Francisco do Prado, Wanderley Lopes de Souza, Sissi Marilia dos Santos Forghieri Pereira, and Luís Ferreira Pires

Abstract

In this paper we propose to apply domain ontologies in agile software development to reduce the ambiguity caused by using natural language as ubiquitous language to report user stories. To justify and demonstrate our approach, we present a case study that combines Scrum and Behaviour-Driven Development (BDD) in the development of an educational support system, which was built to support the activities of the Medicine Programme of Federal University of São Carlos (UFSCar) in Brazil. Starting from a reference ontology for the Higher Education domain, we gradually specialized this ontology for this programme. Since we selected the Evaluation Management module of this system for our case study, we applied the Evaluation Process Ontology to that programme, and defined user stories to identify the feature set to be implemented. For evaluation and validation purposes, we assessed the quality of all ontologies used in this work according to structural and functional dimensions.

Keywords

Ontology · PBL · LMS · Scrum · BDD

37.1 Introduction

Agile software development requires iterative development methodologies, like e.g., Scrum [1], in which requirements and solutions evolve through collaboration between clients and developers. Scrum prescribes *sprint reviews*, which are development team meetings, and the role of *Product Owner (PO)* to plan and evaluate sprints. A list of prioritised requirements, named *product backlog*, is created in a *sprint planning meeting*, and in each sprint the development team decide which requirements should be addressed, and then create a *sprint backlog* that contains the tasks to be performed during that sprint.

Behaviour-Driven Development (BDD) [2] is a development methodology based on the principle that "*stakeholders and developers should refer to the same system in the same way.*" This requires a ubiquitous language understandable by all developers, and that enables executable granular specifications of the system's behaviour and testing.

Ontologies are used in the design of Information Systems (IS) in several domains, with potential benefits due to their mathematical rigour. According to Guarino [3], an ontology can impact an IS both in the *temporal dimension*, depending whether it is used at design time and/or runtime, and in the *structural dimension* when it affects the main IS components.

The Medicine Programme of the Federal University of São Carlos (UFSCar) in Brazil follows a socio-constructivist educational approach, and employs active learning methodologies such as Problem-Based Learning (PBL) [4]. EAMS-CBALM (Educational and Academic Management System for Courses Based on Active Learning Methodologies, [5]) is a system developed to provide computational support for these learning methodologies. EAMS-CBALM was developed using Scrum, and during its development it was often necessary to redefine system behaviour scenarios, and consequently the product backlog items, due to misunderstanding of the stories reported by the PO. In addition, test suites

P. L. de Souza (✉) · A. F. do Prado · W. L. de Souza
Department of Computing, Federal University of São Carlos, São Paulo, Brazil

S. M. dos Santos Forghieri Pereira
Department of Medicine, Federal University of São Carlos, São Paulo, Brazil

L. F. Pires
Department of Electrical Engineering, Mathematics, and Computer Science, University of Twente, Enschede, The Netherlands

S. Latifi (ed.), *Information Technology – New Generations*, Advances in Intelligent Systems and Computing 738,
https://doi.org/10.1007/978-3-319-77028-4_37

defined for this system were incomplete and not always properly covered the system requirements.

Inspired by these problems, we defined the following research questions: (a) "*How to improve communication among PO and developers?*", and (b) "*How to eliminate the ambiguities inherent to natural languages when reporting user stories?*" These questions led to two hypotheses: (1) combining BDD with Scrum can substantially improve communication, and (2) ontologies can eliminate natural language ambiguities. We validated hypothesis (1) in [6] with a case study using EAMS-CBALM, by proposing a ubiquitous language to define BDD scenarios and acceptance tests, allowing the PO to properly communicate with the development team members. This paper focuses on hypothesis (2), by employing domain ontologies as ubiquitous language for software development, and it is further structured as follows: Sect. 37.2 discusses related work; Sect. 37.3 introduces a reference ontology for the Higher Education domain; Sect. 37.4 presents the UFSCar ontology; Sect. 37.5 reports on our case study; and Sect. 37.6 presents some final remarks.

37.2 Related Work

In our systematic literature review of ontologies and agile software methodologies, we found some developments that combine these methodologies to improve software development.

In [7], the OntoSoft agile process is proposed, which 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.

An approach is presented in [8] 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 modelling and on a multi-agent system for providing intelligent assistance to workers.

In [9], the Quality User Story (QUS) framework is proposed 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.

An ontological model is introduced in [10] 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.

All these related developments employ ontologies for specific purposes: for boosting the collaboration among software and ontology engineers, for modelling the Scrum development process, for extracting semantic information to improve user stories, and to support test automation. In our work, in contrast, 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.

37.3 Ontologies

An ontology can be classified according to its generality and dependence levels in: (a) *Top-level ontology*, describes 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, Programme) 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 specializing terms introduced in the top-level ontology; and (d) *Application ontology* describes concepts that depend 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 [3].

A top-level ontology aims at supporting semantic interoperability among domain ontologies. In some situations, domain ontologies may have to be merged, and if they are derived from the same top-level ontology this can be automated. Unfortunately, most of the available domain ontologies are not derived from the same top-level ontology, and different domain perceptions, usage intentions, and languages give rise to incompatible application ontologies in the same domain. To deal with these problems, application ontologies have to be defined based on a reference domain ontology.

Higher Education Reference Ontology (HERO) [11] was defined to provide consensual knowledge of the university domain. HERO describes several aspects of this domain, such as organizational structure, staff, and income. Based on HERO key concepts and according to the higher education areas described by Brazilian National Council for Scientific and Technological Development (CNPq), we extended this reference ontology to describe Brazilian universities, as shown in Fig. 37.1.

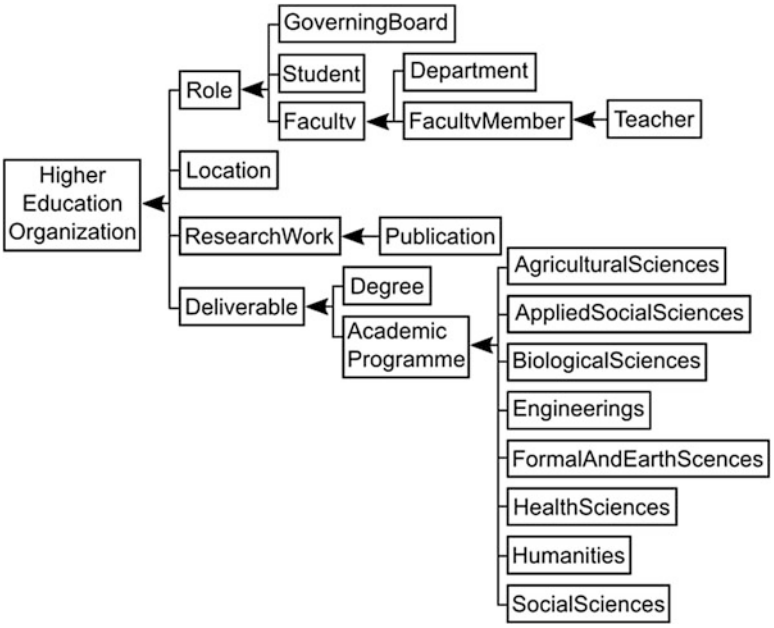

Fig. 37.1 HERO (from [11]) with CNPq areas

37.4 UFSCar Ontologies

We applied the concepts shown in Fig. 37.1 to define an ontology that represents all the UFSCar programmes. Figure 37.2 depicts an excerpt of this ontology in OWLViz/Protégé [12], showing the 8 CNPq large educational areas, and the 10 UFSCar programmes in the Health Sciences area.

37.4.1 UFSCar Medicine Programme

Most Brazilian universities employ teaching-learning methodologies based on classic frontal lectures, but some UFSCar programmes, like the Medicine Programme, employ active learning methodologies. Education in this programme is based on activities with no frontal lectures at all, organized in three educational units: Education Unit of Simulation of Professional Practice (EUSPP), Education Unit of Professional Practice (EUPP), and Education Unit of Elective Activities (EUEA) [13]. Figure 37.3 shows an excerpt of an ontology that describes the UFSCar Medicine Programme.

The UFSCar Medicine Programme learning methodology has several educational activities, such as Problem Situation (PS), Simulation Station (SS) and Team Based Learning (TBL). All of them have learning triggers, which are problems that simulate or portray the daily activities to be performed by the students. Each trigger makes the students traverse the constructivist spiral [13], starting by identifying the problem, formulating explanations, preparing learning questions, looking for new information, building new meanings, and evaluating the process. Figure 37.4 shows an excerpt of the UFSCar Medicine Programme ontology, focusing on its learning methodology, showing some of its educational activities and the steps of the constructivist spiral.

37.5 Case Study

Our case study concentrates on one of the EAMS-CBALM modules. This system was developed using Scrum by the TokenLab commercial company, in collaboration with the Ubiquitous Computing Group (UCG) and Medicine Department (DMed) of UFSCar, and the Teaching and Research Institute (TRI) of the Sírio-Libanês Hospital (SLH).

During this development, weekly sprint meetings were held at TokenLab, and monthly sprint review meetings were held at TRI/SLH. During the TokenLab meetings, the PO (a teacher of the UFSCar Medicine Programme) reported user stories informally in Portuguese, describing activities to be performed by the EAMS-CBALM, and system requirements were also captured and specified in Portuguese. Using these specifications, the developers defined system behaviour scenarios and implemented screen pages, which were discussed at the next meeting. 109 hours have been spent with these meetings, in which the CBALM teaching-learning process was scrutinised, resulting in the definition of the functional and non-functional system requirements, and system architecture [6].

The Evaluation Management module of EAMS-CBALM generated the most controversies between PO and developers. This module had 14 functional requirements and although the scenarios of the "Evaluation Instrument Register" requirement have been redone several times, its final

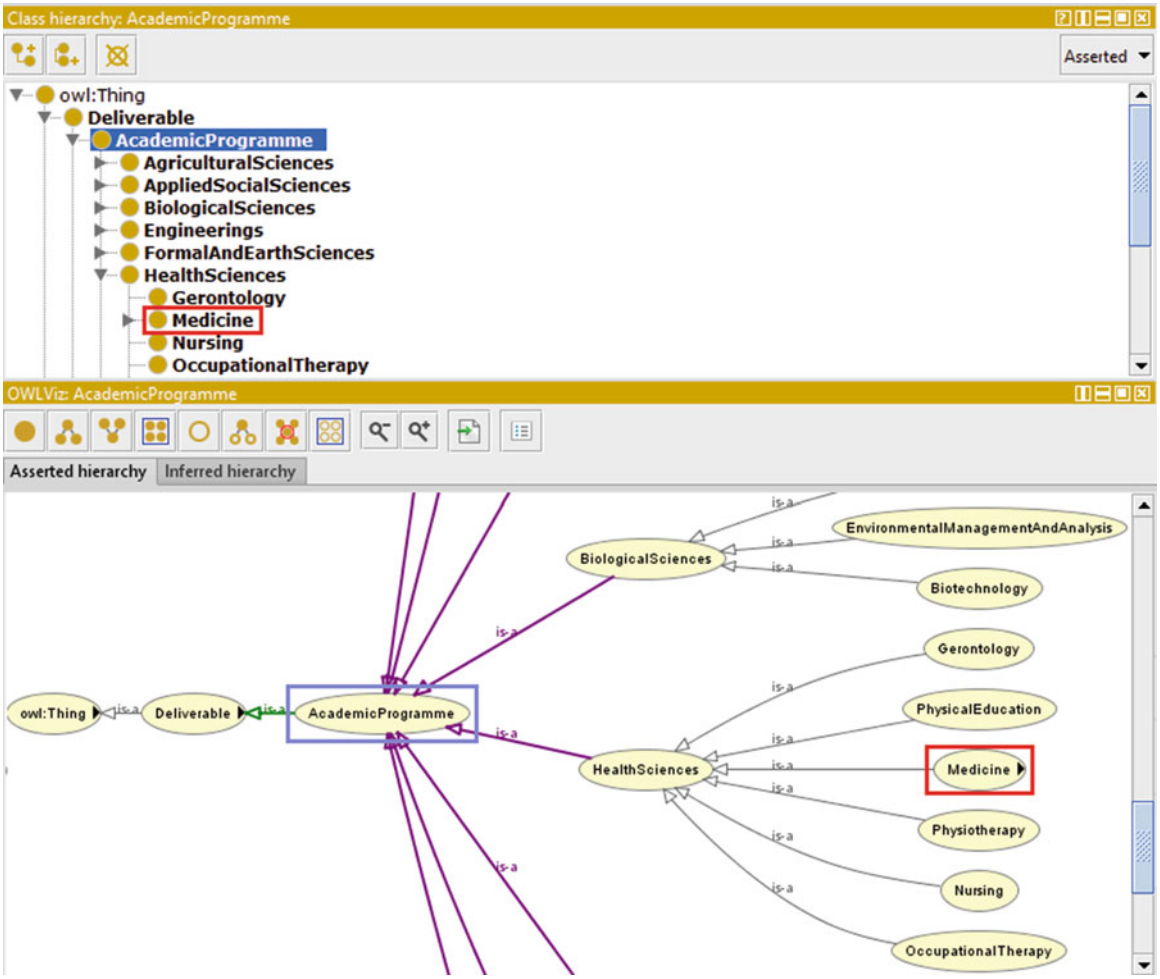

Fig. 37.2 UFSCar ontology (in OWLViz)

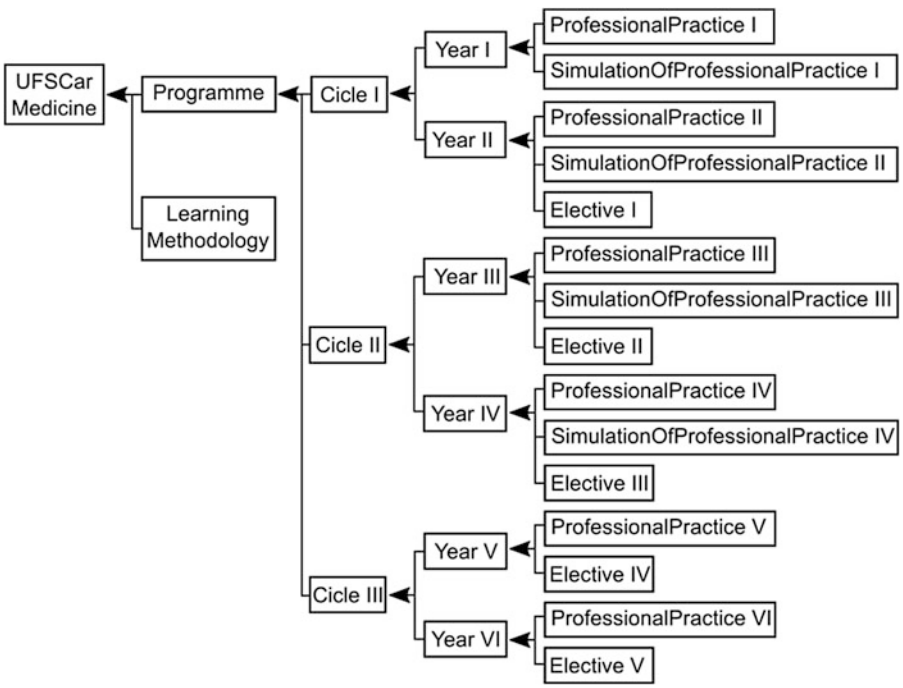

Fig. 37.3 UFSCar Medicine Programme ontology

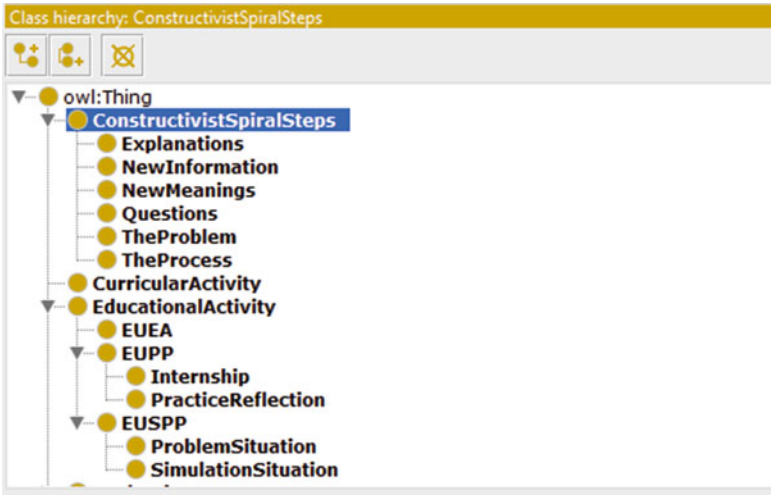

Fig. 37.4 Constructivist spiral steps and educational activities of UFSCar Medicine Programme ontology

implementation did not fully satisfy the PO. This module was already selected in [6] to validate hypothesis (1) and is used here again to validate hypothesis (2).

37.5.1 Evaluation Process Ontology

The student evaluation process of the UFSCar Medicine Programme is quite peculiar and complex. Evaluations are performed by all people involved in the educational activities [13]. There are two evaluation types, formative and summative, and the results are "satisfactory," "unsatisfactory" or "needs improvement." In the latter case, the student must execute an improvement plan proposed by the teacher, and is then re-evaluated. Evaluations are consolidated by applying six instruments types:

(a) Performance Assessment of the Teaching-Learning Process (PATLP): teacher evaluates the student (formative) in three steps, teacher evaluation, classmate evaluation, and improvement plan;
(b) Reflective Portfolio (RP): teacher monitors each student's portfolio (formative), and students' present according to delivery dates (summative);
(c) Written Examination (WE): questions defined by the teacher, answered by the student, and then assigned by the teacher (summative). All questions must have a "satisfactory" result for the student to pass. Failed questions are considered as progress deficit and are worked out in the next WE;
(d) Progress Test (PT): multiple choice questions. Teacher monitors each student's performance (formative) and presence gives students a "satisfactory" result (summative);
(e) Objective and Structured Evaluation of Professional Performance (OSEPP): students act in clinical cases and are evaluated by the teacher similarly to WE (summative);
(f) Problems Based Exercise (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 problem situation (formative).

Based on these instruments and [13], we defined the terminology (names, adjectives, and verbs) to be formally represented in the Evaluation Process. Figure 37.5 shows an excerpt of this ontology, where the *EducationalActivity* and *EvaluationProcess* classes model the concept of *CurricularActivity*.

Each *EducationalActivity* starts with one or more meetings, where a *Meeting* has the participation of students and teachers, each one with specific *Roles*, and triggers a *LearningTrigger*. Each *Learning Trigger* transverse the *ConstrutivistSpiralSteps*, and ends with an *EvaluationProcess*, which is consolidated by applying *EvaluationInstruments*.

Figure 37.6 shows an excerpt of the Evaluation Process ontology of the UFSCar Medicine Programme, focusing on its evaluation instruments types, and showing the PATLP instrument.

37.5.2 User Story and Scenario Ontology

In the BDD analysis phase the most expected system behaviours are identified from the business outcomes to be produced by the system. Based on them, feature sets are defined, where each feature indicates what should be accomplished to

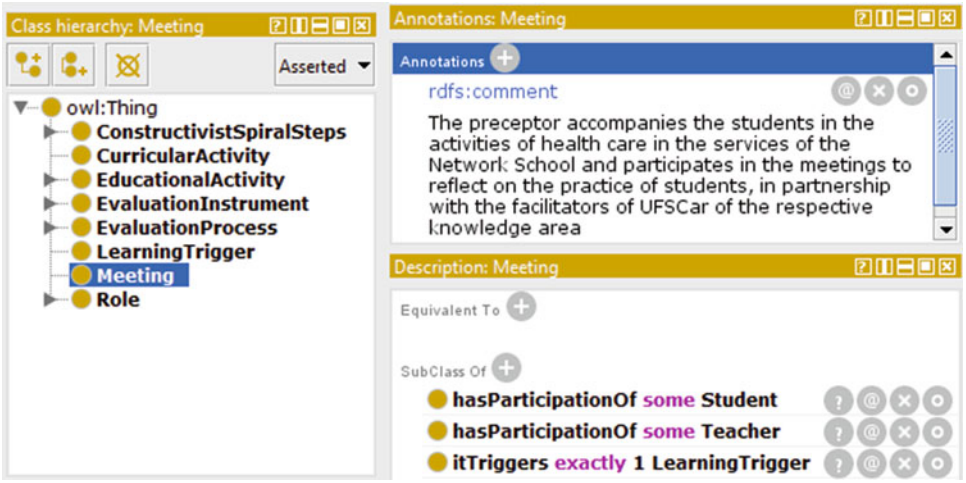

Fig. 37.5 Evaluation Process ontology root classes (*left-side*) and *Meeting* class relations (*right-side*)

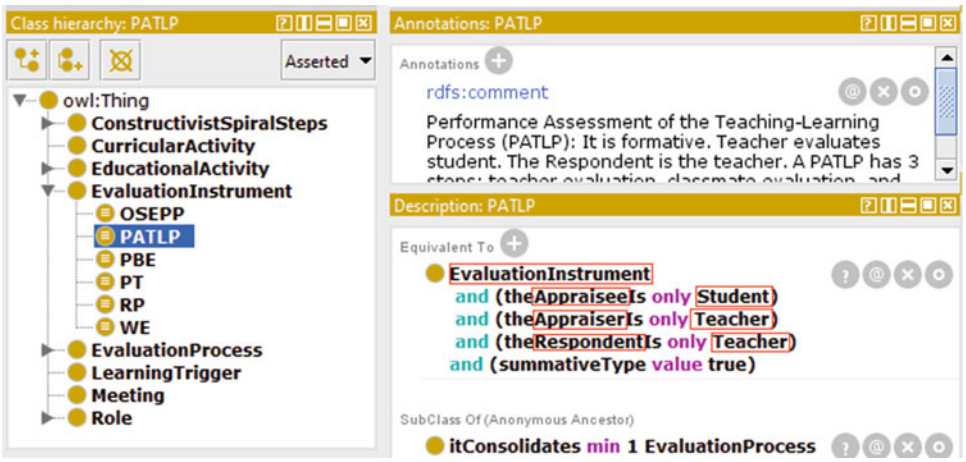

Fig. 37.6 Evaluation Process ontology with PATLP

achieve a specific business outcome. When combining BDD with Scrum, the POs and developers should agree on the feature sets, and ideally define them together.

Features are expressed by user stories, describing 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 (*scenarios*), which in turn should describe specific contexts and outcomes of this user story. For our case study, we have taken into account 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.

Narrative:[*story title*] **In order to**[*benefit*] **As a**[*role*] **I want to**[*feature*]	**Senario:**[*scenario title*] **Given**[*main context*] **And**[*additional contexts*] **When**[*specific event*] **Then**[*main outcome*] **And**[*additional outcome*]

Fig. 37.7 JBehave User Story and Scenario templates

BDD user stories and scenarios follow the templates described in [2], but BDD tools generally do not strictly follow these models. For example, JBehave [14] supports a slightly different user story template and the same scenario template, which are shown in Fig. 37.7. Since JBehave supports most of the BDD characteristics, it is well-accepted in the BDD community, it is open source software, and is frequently updated, we chose this tool to develop the case study reported in [6].

This scenario template is similar to an Extended Finite State Machine (EFSM) model, which was formally defined as an OWL ontology in [10]. We employed a similar EFSM model using the JBehave templates to build the User Story and Scenario ontology for our case study. There are six

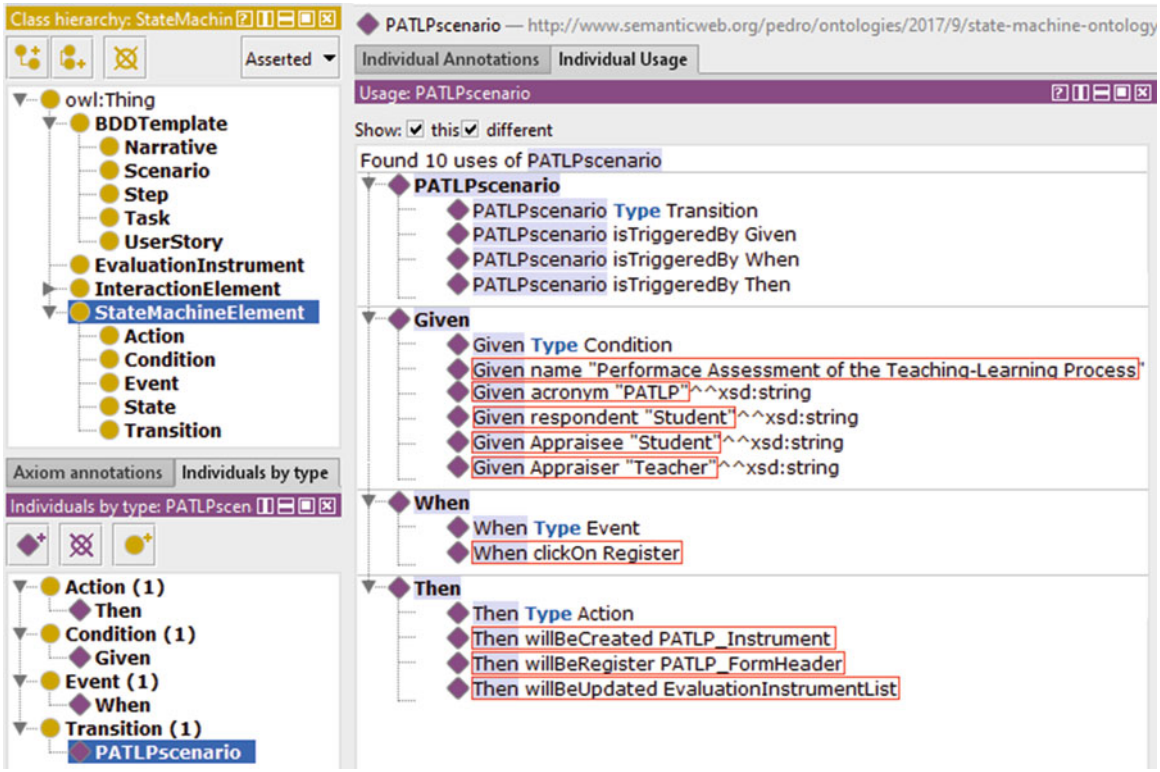

Fig. 37.8 User Story and Scenario ontology with PATLP

scenarios in our case study, one for each instrument type. Figure 37.8 shows an excerpt of this ontology for PATLP.

To validate our approach, we assessed the quality of our ontologies according to the structural and functional dimensions [15]. Structural validation considers the ontology logical structure, focusing on its syntax and formal semantics. Protégé [12] 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.

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.

37.6 Conclusion

In this paper, we proposed the use of domain ontologies to reduce the ambiguity introduced by using natural languages to report user stories. We validated this hypothesis by performing a case study in the context of the EAMS-CBALM using a combination of Scrum and BDD. We started from a reference ontology, and we specialized it for the UFSCar Medicine Programme. We selected a module of this system, and used the Evaluation Process ontology of this programme with the user stories for determining the features to be developed.

This ontology provided the main terminology and its relations to the evaluation process that combined with PO user stories improved the communication between PO and developers. Furthermore, it facilitated the definition of scenarios, as well as the User Story and Scenario ontology.

The ontology presented in [10] is domain-free, and describes behaviours that report steps of scenarios performing actions on the UI. Although it is possible to reuse these steps in multiple testing scenarios, specific business behaviours of our case study had to be specified, and we had to map each term to a user interaction, and to write steps for these interactions. Since this ontology only covers UI testing, we intend to extend it to cover other aspects of software behaviour (e.g., persistence). We will also define a development process that combines Scrum, BDD, and Domain Ontologies.

References

1. K. Schwaber et al., *The Definitive Guide to Scrum: The Rules of the Game* (Scrum.Org and ScrumInc, 2016), 17 pp. https://goo.gl/SBsyUQ

2. D. North, *Introducing BDD* (Dan North & Associates, 2006). https://goo.gl/JBqmry
3. N. Guarino, Formal ontology and information systems. Front. Artif. Intell. Appl. **46**, 03–15 (1998)
4. J. Rhem, in *Problem Based Learning an Introduction*, vol. 8, no. 1 (National Teaching and Learning Forum, 1998), 07 pp. https://goo.gl/LckbVX
5. H.F. Santos et al., Augmented reality approach for knowledge visualization and production in educational and academic management system for courses based on active learning methodologies, in *Proceedings of ITNG 2016,*Advances in Intelligent Systems and Computing, vol. 448 (Springer, 2016), pp. 1113–1123
6. P.L. Souza et al., *Combining Behaviour-Driven Development with Scrum for Software Development in the Education Domain*, vol. 2 (SCITEPRESS–Science and Technology Publications Ltd, 2017), pp. 449–458
7. J.B. Machado et al., OntoSoft Process: Towards an agile process for ontology-based software, in *Proceedings of 49th Hawaii International Conference on System Sciences* (IEEE Computer Society, 2016), pp. 5813–5822
8. Y. Lin et al., Multi-Agent System for intelligent Scrum project management. Integr. Comput. Aided Eng. **22**(3), 281–296 (2015)
9. G. Lucassen et al., Improving agile requirements: the Quality User Story framework and tool. Requir. Eng. **21**(3), 283–403 (2016)
10. T. Silva et al., A behavior-based ontology for supporting automated assessment of interactive systems, in *Proceedings of 11th International Conference on Semantic Computing* (IEEE Computer Society, 2017), pp. 250–257
11. L. Zemmouchi-Ghomari et al., Process of building reference ontology for higher education, in *Proceedings of World Congress on Engineering*, vol. 3 (2013), 06 pp.
12. M. Horridge, *A Practical Guide to Building Owl Ontologies Using Protégé 4.0 and CODE Tools*, Edition 1.1 (University of Maryland, 2007)
13. UFSCar, *Curso de Medicina—CCBS Projeto Político Pedagógico* (Medicina UFSCar, 2007), 139 pp. https://goo.gl/NmWYi3
14. JBehave, *JBehave* (2015). http://jbehave.org/
15. A. Gangemi et al., in *Modelling ontology evaluation and validation*. Lecture Notes in Computer Science, vol. 4011 (2006), pp. 140–154