





Towards a Domain Model for Learning and Teaching

Oleg Shvets¹^a, Kristina Murtazin¹^b, Martijn Meeter²^c and Gunnar Piho¹^d

¹Department of Software Science, Tallinn University of Technology (TalTech), Akadeemia Str 15A, Tallinn, Estonia

²LEARN! Research Institute, Vrije Universiteit Amsterdam, De Boelelaan 1105 1081 HV, Amsterdam, The Netherlands

Keywords: Higher Education, Teaching, Domain Model, Domain Engineering, Model-Driven Architecture, Business Archetypes and Archetype Patterns.

Abstract: In software engineering, a domain model is a conceptual model that represents the concepts and relationships within a particular domain (education, banking, transportation, etc.) by modelling the behaviour and data of these concepts and relationships. This paper presents a novel domain model for learning and teaching in higher educational institutions, addressing the evolving needs of digitized education systems. We integrate concepts from domain engineering, model-driven architecture, and business archetypes to construct a comprehensive framework. Our model, developed by a team of experts from Tallinn University of Technology and Vrije Universiteit Amsterdam, focuses on the interplay between various educational components, including curriculum design, student assessment, and personalized feedback mechanisms. Two educational scenarios – providing feedback on student assignments and managing workplace-based learning – are examined to evaluate the model’s usability. Our contribution lies in offering a structured framework for modelling educational processes, thereby paving the way for more universal and interoperable domain models and ontologies that also support personalized learning and feedback.


1 INTRODUCTION


Digitalization has transformed the way students learn, and teachers teach. It has enabled students to access educational resources from anywhere in the world, at any time, and at their own pace. Digitalization has also made education more inclusive by providing equal opportunities for disabled students. Automating the learning process may revolutionize learning and teaching further by reducing teachers’ workload and providing students with personalized learning experiences and feedback. However, some issues still need to be addressed, such as the lack of digital literacy among teachers and students, the lack of human interaction and the lack of standardization in automated assessments. By addressing these issues, we can ensure that the benefits of digitalization and automation of the learning process are available to everyone.


Domain modelling is a crucial step in the software development process. It involves creating a conceptual model of the problem domain, which helps devel-


opers understand the system’s requirements and design an appropriate solution. It helps developers identify the key concepts, relationships, and constraints relevant to the problem domain (Bjørner, 2006). In healthcare, domain modelling is developing apace. Due to healthcare systems and data interoperability needs, different healthcare domain models exist, including healthcare domain-related standards, e.g. ISO 13940 (ContSys, a system of concepts to support continuity of care) (ISO, 2015; Söerd et al., 2023). In education, however, the situation is different. We only know one standard related to teaching and learning (Educational Organization (ISO, 2018)), and we found only a few papers on Google Scholar related to the educational domain model (e.g. (Sottolare et al., 2016; Brusilovsky, 2016)).

ISO 21001:2018 is the only ISO standard that is directly related to education. It specifies requirements for a management system for educational organizations when such an organization needs to demonstrate its ability to support the acquisition and development of competence through teaching, learning, or research. It also aims to enhance the satisfaction of learners, other beneficiaries, and staff through the effective application of its management system, includ-

^a <https://orcid.org/0000-0003-2171-3512>

^b <https://orcid.org/0000-0001-9605-756X>

^c <https://orcid.org/0000-0002-5112-6717>

^d <https://orcid.org/0000-0003-4488-3389>

ing processes for improvement of the system and assurance of conformity to the requirements of learners and other beneficiaries.

Sottolare et al. (Sottolare et al., 2016) present a conceptual model of intelligent tutoring systems. According to them, these have four main components: the domain model, the student model, the tutoring model, and the user interface model. They use the term 'domain model', unlike the domain model concept outlined in our current position paper, for the ideal expert knowledge, including the bugs, mal-rules, and misconceptions that students periodically exhibit; it contains the set of skills, knowledge, and strategies/tactics of the topic being tutored. The student model contains the cognitive, affective, motivational, and other psychological states students develop during learning. As the monitoring of learner performance is primarily tracked in the domain model, the learner model is viewed as an overlay of the domain model. The tutor or pedagogical model takes the domain and learner models as input and selects tutoring or pedagogical strategies. The user interface interprets the learner's communication and learning contribution through input (how students get information) and output (how students perform assignments) media, e.g. text, diagrams, animations, agents, etc., including natural language interaction like speech recognition and the sensing of learner emotions. Our current position paper's domain model concept should ideally contain all four main components proposed by Sottolare et al.

There are various proposed domain models focusing on some aspects of learning and teaching like life-long learning (Hermans, 2015), qualification-based learning (Then, 2020), curriculum design (Melillán et al., 2023), personalized feedback (Bogdanova and Snoeck, 2019), and learning outcomes (Bogdanova and Snoeck, 2017). Brusilovsky (Brusilovsky, 2016) proposes a personalized guidance domain model for students' monitoring that helps enhance the learning process's overall effectiveness for individual students.

We aim to develop a general and universal domain model for learning and teaching in higher education institutions grounded in the abstract process domain model. This model should not only encompass process objectives and outcomes but should also integrate a robust feedback mechanism. In this context, we are guided by two pivotal research questions: a) Is the creation of such a domain model feasible? and b) Is the implementation of this domain model reasonable and justifiable in terms of its practical applicability and potential benefits to the educational processes and supporting information systems?

This paper proposes a first draft of a simple and

generalized domain model for learning and teaching in higher educational institutions based on the process domain model. The proposed domain model is a conceptual framework that defines and organizes the key elements and concepts related to the teaching process of a specific subject or topic. It provides a basis for describing the learning content in the context of domain knowledge, including the set of skills, knowledge, strategies of the tutored topic, expert knowledge, and misconceptions students may make. It helps organize the education processes and provides personalized management, support, and monitoring of students.

The rest of the paper is structured as follows. In Section 2, we explain how we designed the domain model for learning and teaching. We propose and explain the model in Section 3. In analysis and discussions Section 4, we explain why we designed the model as we designed it. We also discuss two use cases: work-based learning within higher education and evaluating students' assignments. We conclude in Section 5.

2 METHODOLOGY

We follow archetypes and archetype patterns based methodology of domain engineering (Piho, 2011), (Piho et al., 2010a). This methodology is based on the domain engineering methodology proposed by Dines Bjørner (Bjørner, 2006) and utilizes the Zachman Framework (Zachman, 1987), (Piho et al., 2010b) and archetypes and archetype patterns based model-driven architecture (Arlow and Neustadt, 2003), (Piho et al., 2012), (Piho et al., 2011b). The general idea of the methodology is illustrated in Figure 1.

Business requirements					
What	How	Where	Who	When	Why
Things	Processes	Locations	Persons	Events	Strategies
Products and services	Reporting (feedback)	Organization and organization structure	Persons	Business events	Business rules
<i>Product AP</i>	<i>Party relationship AP</i>			<i>Order AP</i>	<i>Inventory AP</i>
<i>Rule AP</i>					
<i>Quantity and money AP</i>					
<i>Common infrastructure</i>					

Figure 1: Zachman Framework with archetypes and archetype patterns.

By asking Zachman Framework questions What, How, Where, Who, When and Why, we design a domain model using concepts of Things, Processes, Locations, Persons, Events, and Strategies by using common predefined models (business archetype patterns, AP) for products (including services), parties

(organizations and persons) and party relationships, orders and inventory, and quantity, money and rules.

There are two main theoretical principles that we utilized when developing the archetypes and archetype patterns we used in domain modelling for learning and teaching: the item-type pattern by (Coad, 1992) and the system's evolving pattern by Oei et al. (Oei et al., 1994). According to the item-type pattern, every concept in our system has a specifiable type. This means, for instance, that every process (e.g., study programme for the students admitted in September 2023) has a process type (e.g., programme curriculum from 2020). According to the system's evolving pattern, every item and its type has Valid-From, Valid-To, Recorded, and Deleted date-time attributes (Söder et al., 2023). The system's evolving pattern enables changes (e.g. a new curriculum version) and auditable error corrections (e.g. some attributes were incorrectly entered into the system) according to the need.

While products, parties, orders, inventory, quantity, money and rules archetype patterns (Piho, 2011) are quite strict models of the same business concepts hinted at by the names of these archetype patterns, the following two things have to be explained in more details: a) how do we model business events with orders and inventory; and b) how do we model business processes with the party relationship archetype pattern.

Column 5 (when, events, Figure 1) describes all business events related to the organization's business processes. Examples of these events are "a new order from a customer", "a plan is ready", "some resource has reached the minimal acceptable limit", or "a new employee is hired". Examples in the context of higher education would be: "a student has been admitted to university", "a student has been enrolled on the course", "a student got his/her exam result", etc. All such business events should be logged. In our domain model, we model all such business events utilizing the order (what happened and when) and the inventory (what has been changed) archetype patterns. With the order archetype pattern, any request to change something in the enterprise's inventory book or in another book/list/registry, such as a student list, has been recorded. This is why we utilize the order archetype not only for the purchase and sale but also for the entry of a specific grade in the student's study book.

Column 2 (how, processes, Figure 1) describes business processes. Examples of business processes are "buying", "selling", "producing", "planning", "servicing", "controlling", "reporting", "transporting", and so on. For modelling business processes, we use the business process archetype pattern (Fig-

ure 3). As described earlier, the process model is described by its elements (process, thread, task, action and outcome) and their types (process type, thread type, task type, action type and outcome type). Our business process model metaphor is a subordinate's report/feedback to a supervisor (e.g., students' homework or assignments presented to the teacher). Therefore, the central part of our model is tasks and task types (inherited from the party relationship and party relationship type, respectively) that specify communication/interaction/relationship between two parties (persons, organizations, or even artificial agents) that are "playing" some roles (e.g. student and teacher) in the particular process or generally in the domain. In our understanding, this metaphor is quite powerful for modelling different kinds of business processes and even business plans, where a plan is nothing more than an expected business process with expected outcomes in the future. We utilized the business process archetype pattern to develop a domain model for learning and teaching in higher educational institutions. We explain this in more detail in the analysis and discussion part in Section 4.

In developing the domain model for learning and teaching in higher educational institutions, we first identified the main concepts (e.g. teacher, student, exam, class, etc.) and their relationships. During this period, we conducted a systematic literature review on work-based learning (Murtazin et al., 2020) and of providing student feedback in e-learning systems (Shvets et al., 2020) and evaluated the adequacy of learning outcomes of the work-based study programmes in regular interaction with stakeholders and completed the preliminary analysis of the student and tutor models. In this way, we got the first draft of the proposed domain model.

The domain model we propose in this position paper (Figure 2) was designed by authors during a four-day workshop within four iterative development cycles, where each cycle lasted a day of intensive teamwork. This workshop exclusively involved the authors of this paper, ensuring a focused and collaborative environment. Each day began with constructive criticism of the previous day's domain model version and ended with a new version of the domain model by the end of the workshop day.

Class diagrams illustrated in Figures 2 and 3 use Visual Studio class diagrams notation instead of standard UML. This is because we utilized Test-Driven Domain Modeling (Piho et al., 2011a) and are developing a domain model that is executable, unit-testable and usable as class libraries in software development, or as Martin Fowler pointed out, the model is usable as an embedded or internal domain-specific language

(Fowler, 2010).

In Visual Studio class diagram notation, the rounded rectangular objects are either classes (as in Figure 2) or interfaces (as in Figure 3). The inheritance relation notion is the same as in standard UML. Two other relations in the figures are one-to-one notation (\rightarrow) and one-to-many notation (\rightarrow). For instance, when, in Figure 2, there is *Programme* \rightarrow *Student* means, that Programme consists of many Students. It does not mean that the Student may be enrolled in only one Programme.

In the current paper, we visualize and explain only classes and relations between the classes and leave the specification of class attributes for future work.

3 A DOMAIN MODEL FOR LEARNING AND TEACHING

Figure 2 visualizes the proposed domain model for learning and teaching in higher educational institutes.

A Programme is a study programme with attributes EnrolledStudents, Curriculum (describing the content students must learn to get a diploma), and Courses (describing the general plan of study subjects taught in a particular study year and semester of the study programme). Thus, the study programme is composed of students admitted to the same curriculum (e.g. Business Information Technology Master's at TalTech) in a given academic year (e.g. in 2023), and these students are studying based on a specific version of the curriculum approved for that academic year, and these admitted students have a study programme plan according to the study courses specified in the Courses attribute.

The Curriculum specifies the content the students are learning during the study programme period. The Curriculum has attributes of Content and ExpectedOutcomes. The Content specifies the course syllabuses the students must learn, and the ExpectedOutcomes specify the competencies (skills, dispositions, and knowledge) the students must gain during the study programme period.

A study Programme normally has many Courses the students must pass during the study programme period. Every Course in the study Programme, like the study Programme itself, has the EnrolledStudents attribute. These are the students who have registered for this Course (e.g. "programming fundamentals") in a given semester (e.g. 2023 fall semester). The content of every Course is specified in the attribute Syllabus. The ExpectedOutcomes of the Syllabus are specifying the Curriculum competencies in detail. In addition, the Course can have one or more course or-

ganizers (professor, lecturer, assistant, etc.; not shown in Figure 2) and one or more Units.

A Unit is a period in the Course (for example, a study week or a study month, as required), and during that Unit, one of the topics (Topic) of the course is addressed. Different activities (Lecture, Lab, Assessment, Project, etc) can be organized during the study Unit, where the Task attribute specifies the content of these activities.

While the Programme and Course are not specific to the particular student (Student) but are specific to a student's group (EnrolledStudents), the Unit and, therefore, the Activity and ActualOutcome, are student-specific. This means that for Unit, Activity and ActualOutcome classes, there is always an attribute Student (not shown in the Figure.2) that specifies the student whose period of study (the Unit) it is, who performed (or performs) the activities (Activity) and who got the actual outcomes (ActualOutcomes).

The Topic is divided into learning Tasks. The Task can be a study instruction (Instruction), Exam, Assignment, etc., and consists of sub-tasks (Content). Sub-tasks can be questions (Question) or any similar that, through possible learning outcomes (PossibleOutcomes), are related and should contribute to the gaining competencies (Competency) and can be used as a check during the learning process. For instance, if the sub-task (SubTask) is some assessment question (Question), the possible outcomes (ResponseCategory) from the learner's point of view is a correct or wrong answer if the question needs a simple "yes" or "no" answer.

A task may have both the entrance and passing rules. For instance, the passing rule for an assessment can be to get at least 6 points out of a maximum of 10, and the entrance rule for the course's final exam can be the requirement of passing all the assessment tasks according to the assessments' passing rules during the same learning unit or all the learning units of the course.

Before explaining the Activity class's Entrance, Passing and Outcomes attributes, we discuss the GradeBook and Grade classes.

Every student has a GradeBook, where all the Grades are recorded. Every Grade is composed of the student's results (Result), and every Result is evaluated by one of the course teachers (not shown in Figure.2). In addition, every Result is related to the Activity (e.g. final exam, project, assessment, etc.) and records the student's actual outcomes for all the sub-tasks related to the Task that describes the content of the Activity. Therefore, in case the activity (Activity) is the assessment (Assessment), and the type of this assessment (Task) is an exam (Exam), then the

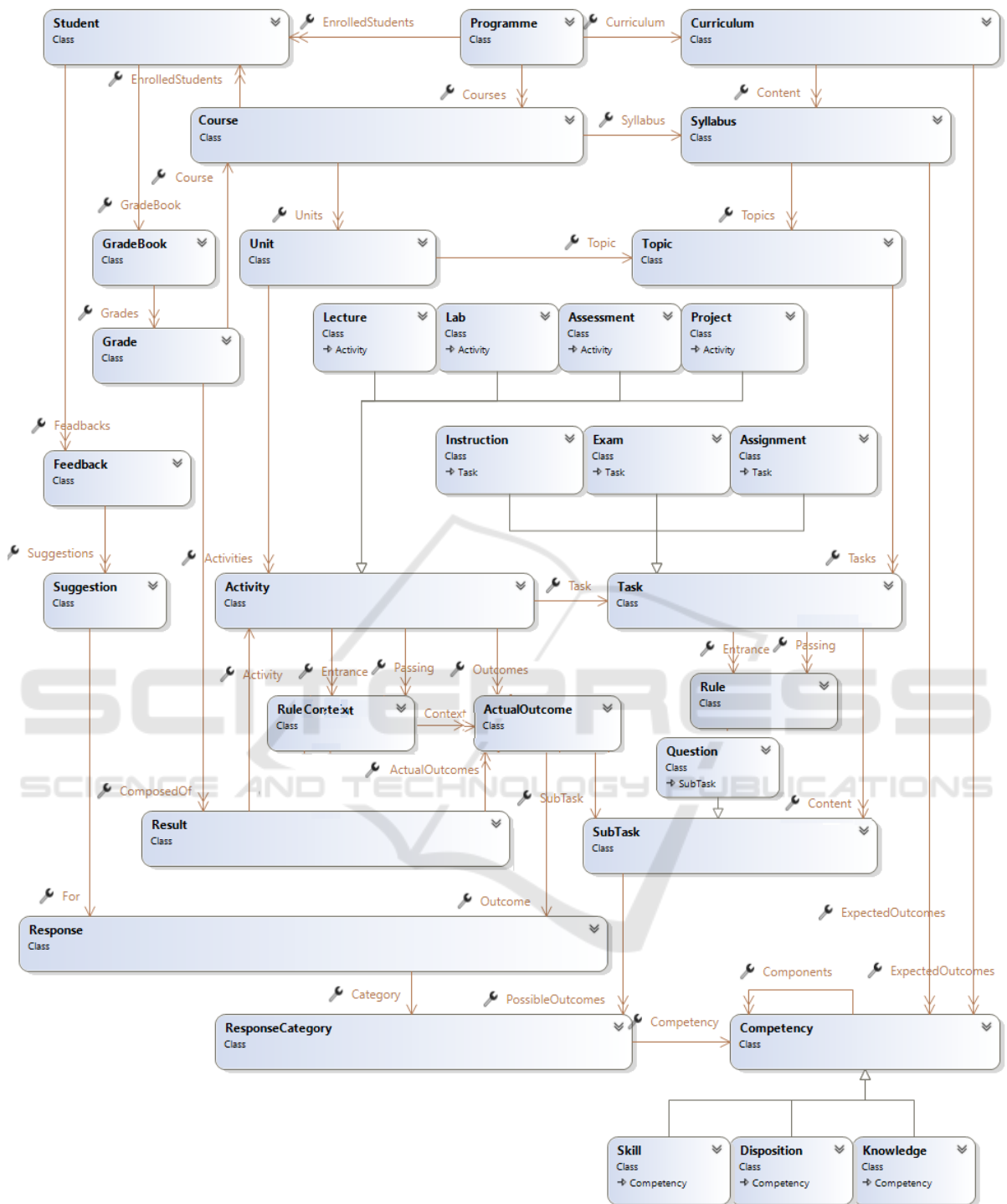


Figure 2: Domain model for learning and teaching.

student’s actual outcomes are the answers to the exam questions (Question), that the evaluator evaluated according to the sub-tasks (Question) possible outcomes (ResponseCategory).

The results thus evaluated (ActualOutcomes) are

used to assign the needed arguments (RuleContext) to calculate the rule value (true/false) of the Task to determine whether a particular student can participate in the given activity or whether he or she has met the conditions required to complete the activity.

These evaluated results (ActualOutcomes) are also used to give a student personalized feedback (Feedback). Such personalized feedback comprises suggestions (Suggestion), where each suggestion can be a predefined response (Response) based on the response category (ResponseCategory) of the student's actual outcome.

4 ANALYSIS AND DISCUSSIONS

Here, we present a preliminary analysis of the proposed domain model. To perfect the model, more detailed analysis and evaluation are needed. Currently, we propose only a rough evaluation by showing three use cases. We first show how a typical university course would be described in the domain model. Then, we explain that work-based learning can also be described using this mode. Finally, we illustrate model compatibility with personalized feedback on students' assignments and homework. In the descriptions, the capitalized words refer to classes in Figure.2.

4.1 A Typical University Course

As an example of a typical university course, the Course "inclusive education" at a large research university consists of lectures, labs and seminars, an exam and a practical assignment. The Syllabus of the course describes two Topics, each covered by one Unit. The first Topic covers theory. Its associated Unit consists of the following Activities: 12 Lectures and one Assessment. The Task within that last Activity is an Exam. This Task contains many SubTasks of class Question, with each ResponseCategory linked to these questions being linked to a Competency that is described in the Syllabus as learning outcomes of the Course.

The second Topic is linked to the competency to write individualized learning plans for students with special educational needs. This Topic is linked to a Unit consisting of the following Activities: Labs and an Assessment. The latter contains a Task that is an Assignment. Its ActualOutcome is graded according to a Rule for passing (formalized in a rubric).

Passing both the Exam and the Assignment creates a Result that is translated into a Grade, which is registered in the GradeBook of an individual Student, helping that Student complete the Course and later the Programme.

4.2 Work-Based Learning

Work-based learning is an educational approach that enhances students' learning experiences by connecting them to real-world work environments. Our domain model is compatible with work-based learning. Work-based learning is often informal, but the formalized learning outcomes could help evaluate the relevance of industry (or company) projects to the university curricula and specify the objectives a student can achieve through the Work-based projects. The following is based on job-based study in the Bachelor of Business Information Technology curriculum of Tallinn University of Technology.

Regarding work-based learning, the course Unit is one study semester (16 weeks) during which students work on a work-based project (Topic). The content of the work-based project, the teaching/learning methods, the course activities/student's reflections, the learning outcomes (ExpectedOutcomes) and the evidence of learning outcomes, assessment criteria and designs are specified in the Syllabus. The work-based learning Activity is a project. The Tasks could be Instruction (in case students need to study some learning materials or read a book during the project), Assessment (the project report, Demo) or Exam (the defence of the conducted project). The Tasks could consist of sub-tasks (Content). Sub-tasks can be questions (Question) or assessment criteria that are related to the competencies (Competency) and can be measured and evaluated through possible learning outcomes (PossibleOutcomes) - how the students can apply knowledge, skills and social or methodological abilities in a work situation. Rules (Rule) are all possible requirements, conditions and prerequisites. ActualOutcome are learning outcomes achieved by one particular student. The Results related to ActualOutcomes could be the personal and professional development plans, learning diaries, learning logs, presentations, interviews or reflective reports, and assessments of learning outcomes, assessing students' achievements according to criteria. The work-based course grade (Grade) comprises the students' results (Result).

4.3 Personalized Feedback

Teacher feedback provided to learners in real-time is crucial for their knowledge and skills acquisition. Feedback (e.g. Corrective feedback) helps learners improve their understanding and performance on various tasks (e.g. exam) (Hattie and Timperley, 2007). However, providing real-time feedback at an individual level is often infeasible, considering limited teach-

performing or solving a task can be divided into very simple categories of responses (ResponseCategory, for example: knows or does not know the Pythagoras formula; knows or does not know what hypotenuse is; can or cannot calculate squares of numbers; etc.), then it should be possible to predefine corresponding feedback responses (Response), based on these categories (ResponseCategory), which are then used to generate automated/semi-automated feedback with suggestions according to the student's actual answer (ActualOutcome).

4.4 The Business Process Archetype

In the context of our proposed domain model for learning and teaching, a significant aspect is its compatibility with the Business Process Archetype Pattern (Figure. 3, (Piho, 2011; Piho et al., 2014)). This compatibility is crucial as it enables the model to align with established business process management frameworks, thus facilitating its integration into existing educational management systems.

The backbone of the model comprises the Programme, Course, Unit, Activity, and ActualOutcome classes. These classes correspond to the Process (Programme), Thread (Course), Task (Unit), Activity (Activity) and Outcome (ActualOutcome) classes in the abstract process archetype pattern. The essence of this compatibility lies in the model's ability to map educational processes into the general business archetype pattern. This mapping allows for a structured approach to managing educational processes, akin to managing business processes.

Moreover, the model's integration with the Business Process Archetype Pattern supports scalability and adaptability. It provides a structured yet flexible framework that can accommodate different educational settings and requirements. This adaptability is particularly beneficial in higher education, where diversity in courses and teaching methodologies is prevalent.

Furthermore, this compatibility ensures that the model can effectively interface with other systems and components of an educational institution, such as administrative systems, learning management systems, and student information systems. It facilitates seamless data flow and interoperability, which are vital for the efficiency and effectiveness of digitalized educational environments.

5 CONCLUSIONS

We proposed a domain model for learning and teaching based on the principles of domain engineering, the Zachman Framework, and archetypes and archetype patterns. This model is designed by educational domain experts, who teach students and develop study programmes and curricula daily. In addition, the model developers in question have sufficient knowledge of modelling techniques and principles. The model was developed over four iterative cycles, with each cycle starting with a criticism of the existing model by analyzing its bottlenecks and ending with a new and improved version of the model. Although a thorough validation and verification of the model is still in progress, a preliminary evaluation of the model has been carried out in this paper by analyzing the model from the ordinary study course, work-based learning perspectives, and from the perspectives of feedback on students' assignments and homework.

The next phase of our research will focus on extensive validation and refinement of the proposed domain model. We plan to further refine the compatibility with the Business Process Archetype Pattern in future iterations of the model. Our aim is to enhance the model's ability to represent educational processes and provide actionable insights for continuous improvement in teaching and learning. This enhancement will involve deeper integration with the Business Process Archetype Pattern, ensuring that the model remains both robust in its theoretical foundation and practical in its application.

Moreover, we aim to conduct empirical studies in diverse educational settings to evaluate the model's practicality and effectiveness in real-world scenarios. This will include testing the model's adaptability to various curricula and its efficacy in providing personalized student feedback. Additionally, we plan to explore the integration of advanced technologies like artificial intelligence and machine learning to further enhance the model's capabilities for automated feedback and adaptive learning strategies. These efforts are expected to yield a more robust and versatile domain model that can significantly contribute to the digital transformation of higher education.

ACKNOWLEDGEMENTS

This work in the 'ICT programme' project was supported by the EU through the European Social Fund. We thank the anonymous evaluators for their valuable feedback.

REFERENCES

- Arlow, J. and Neustadt, I. (2003). *Enterprise patterns and MDA: building better software with archetype patterns and UML*. Addison-Wesley, Object Technology series, Boston, ...
- Bjørner, D. (2006). *Software Engineering 3: Domains, requirements, and software design*. Springer Science & Business Media, Berlin Heidelberg.
- Bogdanova, D. and Snoeck, M. (2017). Domain modelling in bloom: Deciphering how we teach it. In *The Practice of Enterprise Modeling: 10th IFIP WG 8.1. Working Conference, PoEM 2017, Leuven, Belgium, November 22-24, 2017, Proceedings 10*, pages 3–17. Springer.
- Bogdanova, D. and Snoeck, M. (2019). Use of personalized feedback reports in a blended conceptual modelling course. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 672–679. IEEE.
- Brusilovsky, P. (2016). *Domain modeling for personalized guidance*. Army Research Laboratory.
- Coad, P. (1992). Object-oriented patterns. *Communications of the ACM*, 35(9):152–159.
- Deeva, G., Bogdanova, D., Serral, E., Snoeck, M., and De Weerd, J. (2021). A review of automated feedback systems for learners: Classification framework, challenges and opportunities. *Computers & Education*, 162:104094.
- Fowler, M. (2010). *Domain-specific languages*. Addison-Wesley Professional, New York.
- Hattie, J. and Timperley, H. (2007). The power of feedback. *Review of educational research*, 77(1):81–112.
- Hermans, H. (2015). OpenU: design of an integrated system to support lifelong learning.
- ISO (2015). *13940:2015 Health informatics - system of concepts to support continuity of care*. International Organization for Standardization, Geneva. Switzerland.
- ISO (2018). *21001:2018 Educational organizations. Management systems for educational organizations. Requirements with guidance for use*. International Organization for Standardization, Geneva. Switzerland.
- Melillán, A., Cravero, A., and Sepúlveda, S. (2023). Software development and tool support for curriculum design: A systematic mapping study. *Applied Sciences*, 13(13):7711.
- Murtazin, K., Shvets, O., and Piho, G. (2020). Literature review on work-based learning. In *2020 IEEE Frontiers in Education conference (FIE)*, pages 1–8. IEEE.
- Oei, J. H., Proper, H. A., and Falkenberg, E. D. (1994). Evolving information systems: meeting the ever-changing environment. *Information Systems Journal*, 4(3):213–233.
- Piho, G. (2011). *Archetypes based techniques for development of domains, requirements and software: towards LIMS software factory*. PhD thesis, Tallinn University of Technology.
- Piho, G., Roost, M., Perkins, D., and Tepandi, J. (2010a). Towards archetypes-based software development. In Sobh, T. and Elleithy, K., editors, *Innovations in Computing Sciences and Software Engineering*, pages 561–566, Dordrecht. Springer.
- Piho, G., Tepandi, J., Parman, M., Puusep, V., and Roost, M. (2011a). Test driven domain modelling. In *2011 Proceedings of the 34th International Convention MIPRO*, pages 576–581. IEEE.
- Piho, G., Tepandi, J., and Roost, M. (2010b). Domain analysis with archetype patterns based zachman framework for enterprise architecture. In *2010 International Symposium on Information Technology*, volume 3, pages 1351–1356, New York. IEEE.
- Piho, G., Tepandi, J., and Roost, M. (2011b). Evaluation of the archetypes based development. In *Databases and Information Systems VI*, pages 283–295. IOS Press, Amsterdam.
- Piho, G., Tepandi, J., and Roost, M. (2012). Archetypes based techniques for modelling of business domains, requirements and software. In *Information Modelling and Knowledge Bases XXIII*, pages 219–238. IOS Press, Amsterdam.
- Piho, G., Tepandi, J., Thompson, D., Tammer, T., Parman, M., and Puusep, V. (2014). Archetypes based meta-modeling towards evolutionary, dependable and interoperable healthcare information systems. *Procedia Computer Science*, 37:457–464.
- Shvets, O., Murtazin, K., and Piho, G. (2020). Providing feedback for students in e-learning systems: a literature review, based on ieeexplore digital library. In *2020 IEEE global engineering education conference (EDUCON)*, pages 284–289. IEEE.
- Søerd, T., Kankainen, K., Piho, G., Klementi, T., and Ross, P. (2023). Towards specification of medical processes according to international standards and semantic interoperability needs. In *MODELSWARD*, pages 160–167.
- Sottolare, R. A., Graesser, A. C., Hu, X., Olney, A., Nye, B., and Sinatra, A. M. (2016). *Design recommendations for intelligent tutoring systems: Volume 4-domain modeling*, volume 4. US Army Research Laboratory.
- Then, M. (2020). *Supporting qualifications-based learning (QBL) in a higher education institution's IT-infrastructure*. PhD thesis, Dissertation, Hagen, FernUniversität in Hagen, 2020.
- Zachman, J. A. (1987). A framework for information systems architecture. *IBM systems journal*, 26(3):276–292.