Model-Driven Collaborative Design of Professional Education Programmes with Extended Online Whiteboards

Dennis Wolters and Gregor Engels
Department of Computer Science, Paderborn University, Paderborn, Germany

Keywords: Visual Modelling, Instructional Design, Model-Driven, Online Whiteboards, Integration.

Abstract: Designing bespoke professional education programmes with clients and other stakeholders is always challenging. For instance, education providers must understand a client's environment, including the intended audience, and agree with them on learning goals and the programme's structure and content. In this paper, we present our extension to the online whiteboard Miro to collaboratively design professional education programmes with stakeholders in remote settings. We developed a lightweight modelling language covering the entire life cycle of professional education programmes and a visual notation to depict such programmes on a high abstraction level in Miro. Our approach allows the use of online whiteboards beyond informal modelling and makes them a primary tool for designing such programmes. It adds features to extract the underlying model, can deal with modelling flexibility of Miro and provides the basis for integrations with other tools. In particular, we describe the integration with a project management tool to assist with the enactment of design and management processes of professional education programmes. Additionally, we show the feasibility and discuss limitations of tool support for visual modelling languages based on the online whiteboard Miro.

1 INTRODUCTION

Professional education programmes are vital for today's businesses, as they help to attract, develop and retain employees (Kyndt et al., 2009). When companies approach education providers for bespoke professional education programmes addressing their specific needs, programme designers must cooperate closely with stakeholders, such as partners, clients and (potential) participants, to design and deliver high-quality programmes. Models are good mediating artefacts in such situations (Conole, 2009). However, spatial separation and the changed way of working due to the COVID-19 pandemic (Mancl and Fraser, 2020) require this cooperation to be facilitated online, adding additional complexity. Online whiteboards are valuable tools for remote co-design activities (Brandao et al., 2021; Li et al., 2021). Unfortunately, such whiteboards only allow informal modelling. While informal modelling helps to facilitate discussions (Petre, 2013), they make these models temporary artefacts, which have to be transferred manually to other tools for further assistance.

We know from experience the complexity of designing and managing bespoke professional education programmes. Initially, it is necessary to understand a client's environment, e.g. the intended audience or the existing educational resources and tools. Moreover, it must be agreed upon learning goals as well as the structure and content of the programme. Once a programme is developed and running, participants and educators must be managed. Such programmes usually run for multiple iterations and are regularly updated. Designers and managers have to track work items in the respective phases. Project management tools can partially assist in managing tasks and processes, but maintaining consistency and matching task progress against a programme's structure is still done manually.

In this paper, we present an approach that extends the online whiteboard Miro[1] to become a primary modelling tool for facilitating the co-design of professional education programmes. We introduce the Professional Education Programme Modelling Language (PEPML) as a basis for our modelling approach. This language covers the entire life cycle of a professional education programme, from initial analysis to designing to running and adapting a programme. We explain the visual notation and how resulting models can be extracted from Miro boards for further processing, e.g. analysis or integrations with other tools. In particular, we show the integration of our modelling

[1]https://miro.com

Wolters, D. and Engels, G.
Model-Driven Collaborative Design of Professional Education Programmes with Extended Online Whiteboards.
DOI: 10.5220/0011675700003402
In *Proceedings of the 11th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2023)*, pages 133-142
ISBN: 978-989-758-633-0; ISSN: 2184-4348

approach with the project management tool OpenProject[2], which adds additional process management features to our approach. We enable programme designers to ensure that a model representing a programme is kept up-to-date by allowing them to define invariants that must hold when tasks are completed. Also, we show how visual modelling tool support can be created based on Miro and what limitations apply.

This paper is an extended version of (Wolters and Engels, 2022). The previous paper is an extended abstract giving a rough overview of the approach. In this paper, we provide details on PEPML, including its modelling tool support based on Miro, and show how an example can be modelled using our approach.

The remainder of this paper is structured as follows: Section 2 discusses related work. Section 3 describes the metamodel of PEPML and the underlying design goals. Section 4 explains the usage of our approach based on an example professional education programme. Section 5 gives insights into our tool support and its current limitations. Section 6 details the integration with the project management tool OpenProject. Finally, Section 7 concludes the paper and describes future work.

2 RELATED WORK

This section discusses related work from the following areas: (1) Education modelling languages, (2) tool support for designing and managing education programmes, and (3) instructional design processes.

A wide spread of education modelling languages exists (Botturi, 2008). Some of them target the design of education programmes (Paquette et al., 2005; Jurado et al., 2008; Nunes and Schiel, 2014), while others are focussed on learning analytics (Tenorio de Menezes et al., 2017; Auvinen et al., 2014). The IMS-LD specification (IMS Global Learning Consortium, 2003) is an XML-based format to specify online learning programmes. UML Activity Diagrams can be used to visualise IMS-LD specifications, and other visual modelling approaches (Jurado et al., 2008; Paquette et al., 2005) can be converted to IMS-LD-compliant designs. The modelling approach presented in this paper covers all life cycle phases of professional education programmes, which is not the case for all education modelling languages and tools (Nieveen and Gustafson, 1999; Jurado et al., 2008; Auvinen et al., 2014). In contrast to IMS-LD, our modelling language does describe executable models for online learning scenarios. Instead, our approach relates the design and management processes to a programme's description and assists in process enactment. Furthermore, our modelling language provides the basis to integrate tools used in different phases and is intended for use in online whiteboards.

Tool support for instructional design has a long history (Nieveen and Gustafson, 1999). Various IMS-LD-compatible tools have been developed but are no longer maintained (Neumann et al., 2009; Molina et al., 2009). Most commercial tools like Articulate 360 or Adobe Captivate focus on eLearning. The most relevant tool to our work is the education design assistant myScripting (Müller and Erlemann, 2022), a web-based tool that has its strengths in the design phase of online or blended programmes. It also supports collaborative designs but more between multiple educators and less with clients, as it is not directly targeted at bespoke professional education programmes. Moreover, enactment support for design and management processes is not the focus of myScripting.

A broad range of instructional design processes exist (Gagne et al., 2005; Branch and Kopcha, 2014). The most well-known is ADDIE, an acronym for Analysis, Design, Development, Implementation and Evaluation. While ADDIE-based instructional design processes exist that detail each phase (Branch, 2009), we use ADDIE as a description of a professional education programme's general life cycle phases. Allen and Sites (Allen and Sites, 2012) propose an Agile instructional design process called the Successive Approximation Model (SAM). Even though their book title suggests leaving ADDIE for SAM, ADDIE phases can still be observed in SAM. However, SAM emphasizes shorter iterations and prototyping. The instructional engineering method MISA (Paquette et al., 2005) is comprised of a modelling approach and its own design process. Our modelling approach supports the enactment of instructional design processes but stays independent of a specific process by focusing on tasks. Hence, it is compatible with any instructional design process that does not prescribe using a specific modelling language or tool. For instance, it can be used with SAM but not with MISA.

3 PROFESSIONAL EDUCATION PROGRAMME MODELLING LANGUAGE

In this section, we present PEPML's metamodel. We first describe the rationale behind the metamodel and its general structure. Afterwards, we explain the individual packages of the metamodel.

[2]https://openproject.com

3.1 Rationale and Metamodel Structure

Creating a modelling language that covers every relevant aspect of professional education programmes is not feasible. Hence, our approach focuses on a high-level perspective that spans all life cycle phases of a professional education programme and provides extension capabilities to cover additional aspects when necessary. Tackling the low-level perspective, which entails detailed of educational activities, is not advisable as it varies significantly for different types of teaching/learning formats, e.g. planning a Massive Online Open Course (MOOC) is different to a flipped classroom setting. Also, modelling and tool support for specific formats already exists, and off-the-shelf content could be included as well, which does not require detailed planning.

The metamodel's content is based on literature explaining what information is required in the different life cycle phases of instructional design projects (Gagne et al., 2005; Branch, 2009) and our experience in developing bespoke professional education programmes. Most concepts expressed in the metamodel can be found in a similar form in other education modelling languages (Botturi, 2008). A key differentiator to existing education modelling languages is the process enactment support. However, the novelty of our approach lies less in the metamodel itself and more in its utilization. For instance, we use our language to make online whiteboards primary tools for co-designing professional education programmes (see Section 4) and extract PEPML models from these whiteboards to build integrations with other tools (see Sections 5 and 6).

As the colouring in Figure 1 indicates, the metamodel is subdivided into several packages. General concepts relevant to all packages are part of the Foundation package. Five of the packages are named after ADDIE phases. Partially structuring the metamodel based on ADDIE provides a more transparent (internal) structure and shows that PEPML covers the entire life cycle. It is important to empathize that PEPML does not require following an ADDIE-based process. Other instructional design processes such as the SAM (Allen and Sites, 2012) or the Dick & Carrey model (Dick et al., 2005) can be used as well. Concepts from the different packages of the metamodel can be used as needed by the respective design process. The Enactment package provides support for relating processes to a programme and providing assistance in enacting them. Subsequently, the metamodel packages are explained in more detail.

3.2 Metamodel Packages

As the name suggests, the **Foundation** package of PEPML provides the foundation for all other packages. It describes that an education programme consists of entities, which are represented by the class `Entity` or one of its subclasses. The class `Entity` is the superclass of all other classes in the metamodel, which is not explicitly shown in Figure 1 to reduce complexity. Hence, the attributes `name` and `description` are inherited by all other classes. The class `Person` is the superclass for all people involved in a programme. Entities can be enriched with custom properties to ensure extensibility. Categories allow the clustering of entities, and the colour identifies a cluster in the visual notation. We maintain traceability links to previous versions of entities to analyse changes over time. The class `Objective` is the base class for all learning goals and outcomes, which are explained in the two following packages.

The **Analysis** package of the metamodel contains classes to denote analysis results. For instance, it includes two subclasses of `Person`: The class `Persona` is used to represent archetypes of persons attending the programme. The class `Educator` allows specifying people other than participants involved in the programme, such as teachers, facilitators, tutors, subject matter experts or other types of educators. Topics relevant to an education programme, e.g. personal development or IT topics, can be described using the class `Topic`. A hierarchical structure of topics can be defined if needed. Topics can be related to learning goals to define what should be learned with respect to that topic. Learning goals are broader statements about the programme's intended goals. They are not necessarily directly measurable. In that regard, learning outcomes, which are part of the Design package, are more specific than learning goals.

The **Design** package allows describing a programme's structure in terms of education components. An education component is the term we use within PEPML to represent any teaching/learning format, such as instructor-led sessions or project-based learning. The delivery method of education components can range from full face-to-face to blended to fully digital. Education components typically relate to specific objectives, either broad learning goals or measurable learning outcomes. To actually measure a learning outcome, an `EvaluationStrategy` can be defined (see Evaluation package). The class `EducationComponent` is abstract, meaning one of its subtypes has to be used for instantiation. A few teaching/learning formats are defined in the metamodel and additional ones can to be added as required.

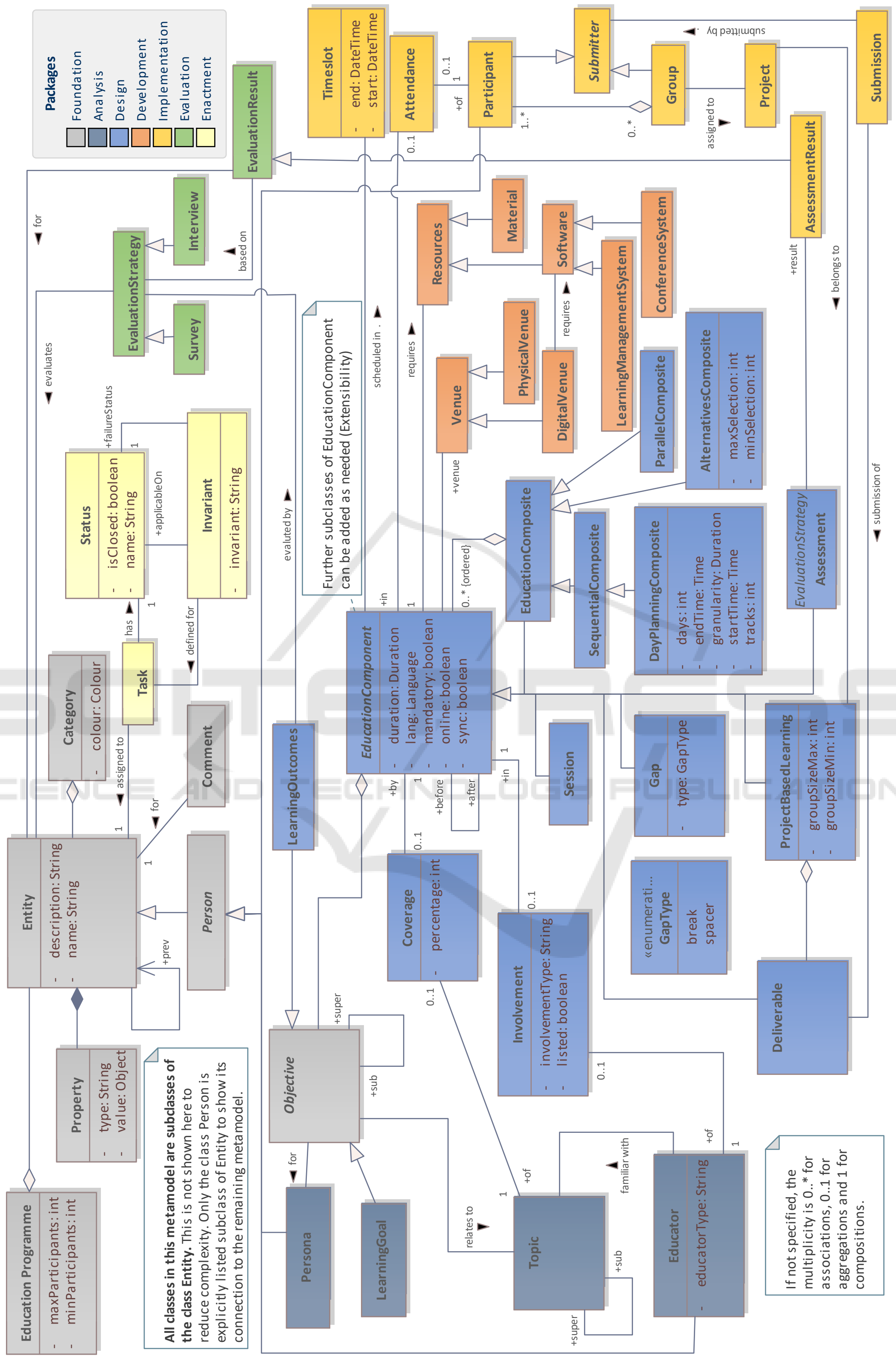

Figure 1: Metamodel of Professional Education Programme Modelling Language (PEPML) depicted as a UML Class Diagram.

Education components are often ordered in a process-like fashion by describing sequences, parallel activities or alternatives (IMS Global Learning Consortium, 2003; Jurado et al., 2008; Nunes and Schiel, 2014). Especially for programmes running over a longer timeframe, components are also aggregated to larger structures scheduled for a given time or dedicated to specific topics. Depending on the context and region, these larger structures are referred to as subjects, modules, courses or units (UNESCO Institute for Statistics, 2011). In this paper, we call them composites to stay agnostic to existing terminology and be independent of a specific context/region. Composites define a temporal order for the components they contain, e.g. components are sequential, run in parallel or are alternatives to one another. The day planning composite can be used to assign components to a specific part of a day. Please note that the day planning composite only assigns a particular time of a day but not when this day will be. This information is further specified in the Implementation package.

The **Development** package entails classes representing venues and resources needed to run the programmes, such as learning materials or software for teaching or communication. Currently, this package is very slim and likely to evolve as we integrate with additional tools, such as learning management systems. Like with any other package, the classes of this package can be relevant in phases than what the package name indicates. We provide an example of this in the next section.

The **Implementation** package contains the classes for representing information on an iteration of an education programme. For instance, it includes classes to capture information about participants, their attendance, projects and submissions. This information will be linked to learning management systems, spreadsheets or file storage tools as part of future work. Furthermore, the package contains the class `Timeslot` to assign concrete timeslots to education components. For instance, it can specify the day represented by a day planning composite. The timeslot is separated from education components to support that a component can be scheduled to different timeslots for different iterations.

The **Evaluation** package defines classes for representing evaluation strategies and their results. Examples of possible evaluations are gathering participant feedback using surveys, interviewing subject matter experts about the content of an education component or using assessments to check if a learning outcome has been reached. This information is likely linked to learning management systems or survey tools.

The **Enactment** package is a differentiator from other education modelling languages as it links a programme to its surrounding processes. It supports process enactment by allowing the assignment of tasks to any entity in a PEPML model. A task is a work item in the context of a specific entity, e.g. a task "Find Lecturer" could be associated with an instructor-led session component. A status describes the state of a task. For instance, the task "Find Lecturer" could have the status "In Progress". The concepts of tasks and statuses are compatible with any process. We use the concepts of tasks and statuses as references to respective concepts in external project management tools. By integrating with such tools, additional information about tasks, such as start/end dates, can be managed externally in the project management tool and does not have to be part of PEPML.

Invariants are key concepts of our approach to keep models consistent and up-to-date. They can be used to define constraints that must be true if a task has a certain status. Further details on invariants and the integration with project management tooling are given in Section 6.

4 MODELLING PROFESSIONAL EDUCATION PROGRAMMES WITH EXTENDED MIRO BOARDS

This section explains how professional education programmes can be co-designed in collaborative whiteboards based on our approach. For this, we introduce a fictional education programme to illustrate the use of PEPML and explain its visual notation. The fictional programme aims to familiarise participants with Agile methods and certify them in one of two Scrum roles. This example programme is meant to show the usage of our approach, and it is not essential if the programme makes sense from a content or didactical point of view.

The following shows the programme from both of PEPML's viewpoints, the dependency and the temporal structure viewpoint, and explains the different modelling options. We do not reflect all concepts of the metamodel in the visual notation since we believe that certain information is better managed in other tools. For instance, information about participants, their attendance and submission can be managed using spreadsheets, file storage solutions or learning management systems. This information can be extracted from these tools and reflected in a PEPML model for further analysis.

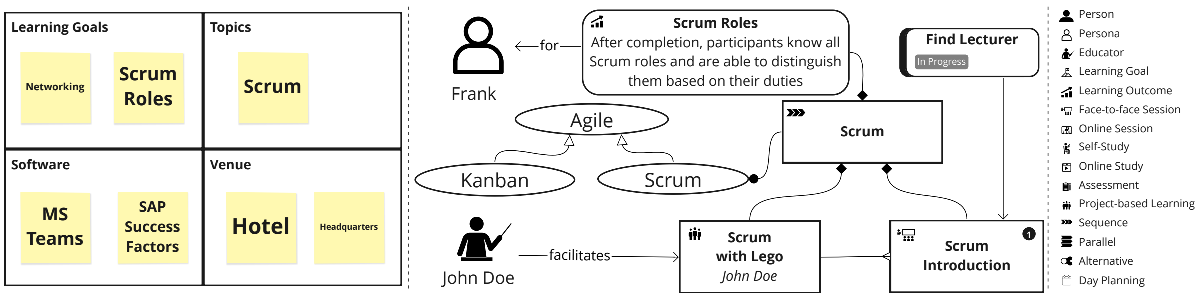

Figure 2: The left part shows how PEPML entities can be created by placing items on specific regions, the middle shows how entities and their dependencies can by expressed as a graph and the right shows optional decorators to denote subtypes.

4.1 Modelling Entities & Dependencies

The first of PEPML's viewpoints is concerned with modelling programme entities and their dependencies. Modelling dependencies is typically done across the phases Analysis, Design and Development. Designers, clients and educators can collaboratively model entities and their dependencies using PEPML's visual notation. We explain the options to depict PEPML entities shown in Figure 2 in this subsection.

In the analysis phase, programme designers and their clients capture essential important information like learning goals, topics to be covered or available resources. For this, regions on the board can be defined using shapes to collect specific information. An example is shown in the left part of Figure 2, where we define different regions for learning goals, topics, available software and venue options. Users can declare that a shape represents a region for a specific PEPML class. Clients and designers can place sticky notes on these regions to create objects of the respective classes. The example shows that elements of other packages can be relevant in the analysis phase since the classes `Software` and `Venue` from the Development package are used to collect available software and potential venues. The modelling technique utilizing specific board regions to denote objects of a specific type can be used in any phase, e.g. to develop a session with an instructor. Miro encourages this modelling technique in their tutorial videos and predefined templates. We know from workshops with dozens of professionals that this way of modelling is intuitive and does not require a long introduction.

Instead of creating PEPML entities by placing them in specific regions, shapes can be used to define the type of a board item (see the middle part of Figure 2). PEPML represents persons and their subtypes, persona and educator, using icons with a label beneath containing the entity's name. Learning goals and outcomes are represented as rounded rectangles, topics as ellipsis and education components as rectangles. We use Miro's app cards to represent tasks, which are visualised as rectangles with bold, left borders. Furthermore, we allow the expression of dependencies between entities using associations.

For our fictional example, we define a persona, Frank, for which a learning goal is defined that relates to the topic "Agile". For the subtopic Scrum, a sequential composite contains two components: a face-to-face session introducing Scrum and an exercise to experience the Scrum process with Lego facilitated by John Doe. The information that the educator "John Doe" is facilitating this exercise is denoted using the association class `Involvement`. A learning goal of the Scrum introduction shall be that the participants afterwards know all Scrum roles.

The metamodel defines various associations between classes. To make the type of association more explicit, we suggest a styling of associations similar to the UML. The reasoning behind this is that users already familiar with the UML can reuse their knowledge to interpret these models. The *"specialization"* association (—▷) is used for hierarchies of objectives and topics. The *"covers"* association (—●) represents the association class `Coverage` between education components and topics. In the example, the Scrum composite covers the Scrum topic. The *"is part of"* association (—◆) depicts compositions and aggregations defined in the metamodel. In the dependency viewpoint, the association "is part of" is used to express that a component is part of a composite. The *"before-after"* association (⋗—) declares a temporal dependency between education components. In the example, the "Scrum Introduction" must be before "Scrum with Lego". This information is exploited when the components are scheduled. The *"relates to"* association (⟶) is used to denote all remaining associations specified in the metamodel, including those set by the association classes `Involvement` and `Attendance`. The label of associations is either defined by the association name in the metamodel or in the case of the association class `Involvement` by the value of the attribute `involvementType`.

The suggested styling of associations does not necessarily need to be used. In some cases, the type of association can be deduced from the connected board items, and we can automatically apply the suggested styling upon request. More information on this is provided in Section 5.

The right part of Figure 2 features a set of optional decorator icons that can be used to refine the type of an entity, e.g. to differentiate between a learning goal and an outcome. These icons can be placed in the top-left corner of entities.

In the top-right corner of entities, we use decorators to indicate the number of associated open tasks. For instance, the ❶ on the "Scrum Introduction" component indicates one associated task: the "Find Lecturer" task. These decorators denoting the number of associated tasks provide an indication which entities have open tasks. The status of a task and other information available based on the integration with the external project management tools are shown as boxes at the bottom of the task's rectangular representation, e.g. due dates or assignees. Section 6 provides more information on associated tasks.

4.2 Modelling Temporal Structure

PEPML features a second viewpoint to denote the temporal distribution of education components across the course of a programme. The viewpoint is relevant to all stakeholders to understand, communicate and discuss a programme's structure. Designers can use this viewpoint to adapt a programme's structure in collaboration with their clients. The centre of Figure 3 shows the fictional example from the temporal viewpoint alongside some of our additions to Miro boards on the left and hinting the model extraction on the right. The latter is further explained in Section 5.

The temporal viewpoint solely focuses on the temporal relations of education components, and shapes other than rectangles are not ignored. While graphs often specify temporal relations in education modelling languages (Auvinen et al., 2014; Jurado et al., 2008; Nunes and Schiel, 2014), PEPML uses vertical and horizontal placement to denote a temporal order. By default, horizontal placement defines sequential ordering, and vertical placement defines parallelism. Alternatives are expressed as optional parallel components, which have a dashed border. Only for the day-planning composite, the vertical axis reflects time since this visualisation is common practice in calendar applications. The representation of a day-planning composite is also the only case where the vertical size of an item has semantic meaning, i.e. it represents the duration.

The example programme shown in the middle of Figure 3 has a top-level sequential composite consisting of three sub-composites. The example programme starts with a two-day kick-off event specified using a day planning composite. Complex board items like the two-day kick-off of the example programme can be generated using our extension (see ❶). Simultaneously, we still allow the use of standard board items. For instance, a rectangle representing a "Scrum with Lego" group activity is placed on top of the generated day planning composite (see ❷). The placement of the component on the day planning composite and its vertical size indicate when this activity is held. The kick-off is followed by a self-learning period consisting of two parallel streams: peer reflection and preparing for one out of two Scrum roles. The latter is specified using an alternative composite, and the dashed border of the contained components indicates that they are optional. The number in the icon of the alternative composite defines how many of the alternatives must be chosen. Hence, even though those components are optional, participants must choose one of them. The programme ends with a two-track closing day with a joint welcome session, followed by separate assessments for the different roles and finishes with handing out certificates. The colouring indicates that entities belong to the same category, e.g. administrative components (blue) or Scrum content (green).

5 TOOL SUPPORT

The tool support for PEPML is based on the collaborative online whiteboard Miro, a Node.js backend and a Neo4j graph database for persistence and querying capabilities. We decided to use Miro over other collaborative online whiteboards because of its popularity, sophisticated API, good documentation and active developer community. We decided against web-based modelling frameworks like Sirius Web or diagram-js because online whiteboards provide more flexibility and are known by a wider audience of end users. Furthermore, we used a subset of PEPML's notation already informally in Miro to plan and discuss professional education programmes in remote settings. However, this informal usage is limited since no model validation or further processing is possible. In the following, we explain how we extract models from Miro boards to overcome these limitations, how these models are persisted in a Neo4j database and what limitations exist when using Miro to develop modelling tool support. The integration with project management tools is discussed in the next section.

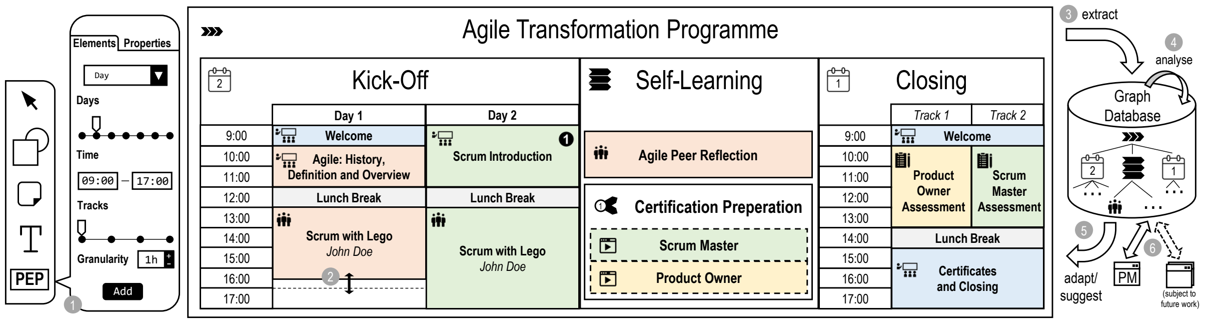

Figure 3: The left part shows parts of our extensions to the online whiteboard Miro, the middle depicts the temporal structure of the example programme, and the right gives an overview of the model extraction.

5.1 Extract, Persist, Analyse, Adapt and Integrate

Our tooling tracks items on Miro boards that represent PEPML entities and uses this information to extract the underlying model (see ❸). Items added to a Miro board via our toolbar extension are automatically tracked as PEPML entities. When items are added using Miro's standard toolbar, they are tracked when added to a frame defined to contain a PEPML model or if they are sticky notes placed on a region representing a specific PEPML class (see left part of Figure 2). Alternatively, users can manually define that an item represents a PEPML entity. Board items not being tracked are ignored during extraction.

The extraction can deal with imprecise positioning by using the centre of board items to determine containment relations. We also allow items to overlap up to a certain threshold. As mentioned in Section 4, association types can be deduced in some cases. For instance, connecting an educator with an education component represents an involvement. In other cases, a direction is needed to properly make sense of an association type, e.g. an undirected association between two topics does not disclose the super/sub-topic. If an association type is deduced, we allow to automatically apply the suggested styling. We ask users to refine the association if the type cannot be deduced.

Composite types only have to be modelled explicitly in case of day planning composite or if they shall be named or their properties need to be manipulated. Otherwise, placing two components in sequence or parallel to one another suffices to deduce a sequential or parallel composite. If conflicts during the extraction occur, users are involved in fixing them.

We allow a PEPML entity to be represented by multiple items on a Miro board. Items represent the same entity if their name is equal. Alternatively, users can manually define that items represent the same entity. The information specified by two items representing the same entity can differ to the extent that additional information is defined. In Figure 2, the learning goal "Scrum Roles" is represented as a sticky note and as a rounded rectangle. The latter contains additional information, such as a description and dependencies to other entities. Two representations of the same entity are not allowed to specify conflicting information, e.g. the items are not allowed to denote that they belong to different composites.

The extracted model is persisted in a Neo4j graph database. Classes of the PEPML metamodel are represented as node labels, and their instances as nodes with the respective labels. Class attributes and additional properties defined for entities are mapped to properties of the respective nodes. By persisting PEPML models in Neo4j, models become queryable using Neo4j querying language Cypher, which allows further analysis and extracting (aggregated) information from models (see ❹). For instance, it can be checked if "before-after" constraints are satisfied in the temporal structure. Our tooling can adapt the visual model on the Miro board by adding type decorators, apply the suggested visual syntax or resolve problems (see ❺). The extracted models can be utilized to integrate our approach with other tools (see ❻). In Section 6, we explain the integration with a project management tool, and we plan to develop additional integrations in the future.

5.2 Limitations

While developing our modelling tool support based on Miro, we discovered certain limitations. We discuss technical limitations that affect development and impact user experience in the following. We plan a user study to gather further insights on user experience, but this is out of this paper's scope.

Miro offers two options to programmatically interact with Miro boards: A Web SDK and a REST API. Unfortunately, they are not equally powerful. The SDK and API overlap partially in their functionality, but there are features exclusive to each of them, which required us to use both for our implementation. For instance, reading or manipulating associations can only be done using the REST API. In contrast, the Web SDK is faster at interacting with a board and can react to item selections on Miro boards. If the Web SDK would allow the same board manipulations as the API, the implementation could be simplified.

Neither the API nor the SDK currently supports bulk operations, meaning board items can only be manipulated one by one, which is noticeably faster with the Web SDK than with the REST API. Support for bulk operation in the API is currently being proposed as a draft and will be released soon. Once bulk operations exist, they will accelerate board manipulations.

The Miro user interface offers a convenient feature to group board items, which does neither exist in the SDK nor the API. The grouping feature is handy for creating complex items like an icon with the text field beneath to denote persons or day planning composite. The feature is on the API/SDK's roadmap; until then, users must group them manually.

Since the grouping feature is not programmatically accessible, we cannot attach icon decorators that denote (sub)types to the corresponding board items. Users can do that manually, but changing the size of the grouped items can lead to decorators that are out of proportion or larger than intended. Miro already has a feature of defining parent-child relations between items and positioning children relative to their parents, which would solve this issue. However, this feature is limited to frames being used as parents. Frames are board items that define regions in Miro boards, which are used for presentations or to export board content as images or PDFs. They cannot be used to substitute other shapes as they cannot be associated with other items or styled like shapes. Until the parent-child feature is extended to other shapes, we can offer to reset decorators programmatically.

Miro supports a range of different shapes. The paid version of Miro includes special shapes for specific modelling languages, such as UML class or BPMN diagrams. However, only standard shapes can be manipulated via the SDK and API. Hence, visual languages can only be based on standard shapes.

6 INTEGRATION WITH PROJECT MANAGEMENT TOOLS

Programme designers and managers must keep track of all tasks required to design and run an education programme. Process management based on tasks works in combination with any concrete instructional design or management process. Instead of adding additional capabilities for task management to PEPML and our tooling, it was our goal to leverage the power of existing project management tools and integrate them with our approach. The Enactment package explained in Section 3 provides the basis for this integration. As a proof-of-concept, we decided to integrate our tooling with the project management tool OpenProject because it is open-source and has a powerful, well-documented API, which we use to synchronize tasks and statuses between PEPML models and OpenProject. In the following, we provide further information on this integration.

OpenProject supports the definition of projects containing different types of work packages. A professional education programme represented by a PEPML model becomes a project within OpenProject. Every entity associated with a task in a PEPML model is added as a feature to a project. Tasks associated with that entity are added as subtasks of the respective feature in OpenProject. In OpenProject, these tasks can be managed using Gantt Charts, lists or Kanban Boards, and additional properties like assignees or due dates can be defined.

As mentioned in the description of the Enactment package, users can define invariants for tasks that must hold if a task is in a specific status. These invariants can ensure that changes resulting from completing tasks are reflected in the model. Invariants are formulated as Cypher queries evaluated on the persisted model in the Neo4j database. An invariant is true if the result of the query is not empty.

A webhook defined in OpenProject informs the PEPML tool about all status changes. When a status changes, the corresponding invariants are evaluated. For instance, we can define an invariant for the status "Closed" of the task "Find Lecturer" shown Figure 2 that requires a lecturer to be set for the "Scrum Introduction". If a user closes this task without setting the lecturer in the model, the invariant would not be satisfied. In that case, the task's status would be changed to a predefined failure status. This example illustrates how invariants define task completion criteria and ensure that models are kept up-to-date.

7 CONCLUSION AND FUTURE WORK

Our modelling approach presented in this paper makes the online whiteboard Miro a primary tool for co-designing professional education programmes in remote settings. Miro's collaborative and flexible modelling is supplemented by extracting the underlying model for further analysis or processing in other tools. Our approach focuses on a high-level view of professional education programmes by modelling programme entities, their dependencies and temporal order. It assists in aligning the understanding of a professional education programme's design between different stakeholders and coordinating design and management activities. Our modelling approach supports process enactment by allowing users to define tasks for any part of a programme and by integrating with project management tooling. Task invariants can be defined to ensure that models stay consistent and up-to-date. While there are minor limitations, our tooling shows the feasibility of using Miro for building tool support for visual modelling languages.

For future work, we plan to conduct a user study to evaluate PEPML and our tooling. Simultaneously, we plan to develop integrations with additional tools and use PEPML as the basis for a situational method engineering approach for instructional design.

REFERENCES

Allen, M. W. and Sites, R. (2012). *Leaving ADDIE for SAM*. American Society for Training and Development.

Auvinen, T., Paavola, J., and Hartikainen, J. (2014). STOPS: A Graph-based Study Planning and Curriculum Development Tool. In *Koli Calling '14*, pages 25–34. ACM.

Botturi, L. (2008). *Handbook of Visual Languages for Instructional Design: Theories and Practices*. Information Science Reference.

Branch, R. M. (2009). *Instructional Design: The ADDIE Approach*. Springer.

Branch, R. M. and Kopcha, T. J. (2014). Instructional Design Models. In *Handbook of Research on Educational Communications and Technology*, pages 77–87. Springer.

Brandao, E., Adelfio, M., Hagy, S., and Thuvander, L. (2021). Collaborative Pedagogy for Co-creation and Community Outreach: An Experience from Architectural Education in Social Inclusion Using the Miro Tool. In *Advances in Human Dynamics for the Development of Contemporary Societies*, volume 277 of *LNNS*, pages 118–126. Springer.

Conole, G. (2009). The Role of Mediating Artefacts in Learning Design. In *Handbook of Research on Learning Design and Learning Objects: Issues, Applications, and Technologies*, pages 188–208. IGI Global.

Dick, W., Carey, L., and Carey, J. O. (2005). *The Systematic Design of Instruction*. Pearson Education Ltd.

Gagne, R. M., Wager, W. W., Golas, K. C., Keller, J. M., and Russell, J. D. (2005). *Principles of Instructional Design*. Wiley Online Library.

IMS Global Learning Consortium (2003). Learning Design Specification Version 1.

Jurado, F., Molina, A. I., Giraldo, W. J., Redondo, M. A., and Ortega, M. (2008). Using CIAN for Specifying Collaborative Scripts in Learning Design. In *Cooperative Design, Visualization, and Engineering*, volume 5220 of *LNCS*, pages 204–211. Springer.

Kyndt, E., Dochy, F., Michielsen, M., and Moeyaert, B. (2009). Employee Retention: Organisational and Personal Perspectives. *Vocations and Learning*, 2(3):195–215.

Li, Q., Zhang, J., Xie, X., and Luximon, Y. (2021). How Shared Online Whiteboard Supports Online Collaborative Design Activities: A Social Interaction Perspective. In *Advances in Creativity, Innovation, Entrepreneurship and Communication of Design*, volume 1218 of *AISC*, pages 285–293. Springer.

Mancl, D. and Fraser, S. D. (2020). COVID-19's Influence on the Future of Agile. In *Agile Processes in Software Engineering and Extreme Programming – Workshops*, volume 396 of *LNBIP*, pages 309–316. Springer.

Molina, A. I., Jurado, F., de la Cruz, I., Redondo, M. A., and Ortega, M. (2009). Tools to Support the Design, Execution and Visualization of Instructional Designs. In *Cooperative Design, Visualization, and Engineering*, volume 5738 of *LNCS*, pages 232–235. Springer.

Müller, C. and Erlemann, J. (2022). Educational Design for Digital Learning with myScripting. In *EDEN Conference'22*, pages 128–134.

Neumann, S., Oberhuemer, P., and Derntl, M. (2009). Visualizing Learning Designs Using IMS Learning Design: The Position of the Graphical Learning Modeller. In *ICALT'09*, pages 732–733. IEEE.

Nieveen, N. and Gustafson, K. (1999). Characteristics of Computer-based Tools for Education and Training Development: An Introduction. In *Design Approaches and Tools in Education and Training*, pages 155–174. Springer.

Nunes, I. D. and Schiel, U. (2014). Using High Level Activities Net for Learning Analytics of Instructional Design. In *ICALT'14*, pages 383–385. IEEE.

Paquette, G., de la Teja, I., Léonard, M., Lundgren-Cayrol, K., and Marino, O. (2005). An Instructional Engineering Method and Tool for the Design of Units of Learning. In *Learning Design: A Handbook on Modelling and Delivering Networked Education and Training*, pages 161–184. Springer.

Petre, M. (2013). UML in Practice. In *ICSE '13*, pages 722–731. IEEE.

Tenorio de Menezes, D. A., Florencio, D. L., Silva, R. E. D., Nunes, I. D., Schiel, U., and de Aquino, M. S. (2017). DaVID — A Model of Data Visualization for the Instructional Design. In *ICALT'17*, pages 281–285. IEEE.

UNESCO Institute for Statistics (2011). International standard classification of education: ISCED 2011.

Wolters, D. and Engels, G. (2022). Model-driven Design and Management of Professional Education Programmes. In *ICSOB'22 Companion Proceedings*, CEUR Workshop Proceedings. (in press).