Modeling and Mining of Learnflows

Robin Bergenthum[1], Jörg Desel[1], Andreas Harrer[2], and Sebastian Mauser[1]

[1] FernUniversität in Hagen, Lehrgebiet Softwaretechnik und Theorie der Programmierung
forename.surname@fernuni-hagen.de
[2] Katholische Universität Eichstätt-Ingolstadt, Fachgebiet Informatik
andreas.harrer@ku-eichstaett.de

Abstract. This article transfers concepts and methods from business process modeling and workflow management to the field of learnflows, i.e. learning and teaching processes. It is first shown that these two areas have a lot of commonalities and similarities. On the other hand, there are also crucial specifics of learning processes that have to be taken into account additionally. We then introduce and discuss modeling languages for learnflows which are based on ideas from workflow modeling. Finally, we develop an approach to automatically generate learnflow models from log files of learning systems by adapting workflow mining methods.

1 Introduction

Learners' activity is in the focus of modern strands in pedagogy, such as constructivistic and self-regulated learning. Research results show that in completely unguided learning situations productive activities like reflection and elaboration rarely happen or are not performed successfully [1]. In contrast, structuring of learning activities by means of scripts [2,3] and scaffolds has proven to be beneficial to learning outcomes. In most of today's learning support systems (e.g. [4]) this type of support is hard-wired and tied to a specific learning domain and system, especially when tightly integrated with the graphical user interface of the system. The learning process models are given only implicitly by the respective systems. This way, re-usability and transferability to other contexts and learning platforms is restricted.

Explicit representation of learning process models and scaffolds is one way to make pedagogical expertise and practice re-usable, thus reducing the effort to develop educational support systems while also stressing the underlying pedagogical design principles. This holds in particular for computer-supported collaborative learning (CSCL) systems. CSCL research investigates in the affordances and effects of computer applications supporting groups of students in knowledge construction and skill development.

While the initiatives to make learning processes more explicit have recently gained scientific prominence under the terms educational modeling [5] and learning design [6], business process modeling or workflow engineering is a well-established research field. There is a large repertoire of mature methods, rigorous

K. Jensen, S. Donatelli, and J. Kleijn (Eds.): ToPNoC V, LNCS 6900, pp. 22–50, 2012.

procedures, and formal approaches that are used for definition, re-engineering, and automatic support of business processes in industry and companies [7,8,9].

Business process modeling and modeling of learning processes obviously share similar traits. We discuss in this work the similarities and differences. We also investigate potential transfer of results and methods from business process modeling to learning process modeling, or shortly from workflows to learnflows. More precisely, we first develop modeling techniques and methods for learnflows that take up approaches from workflow methodology. Then, we present specific procedures and schemas for (semi-)automated synthesis of learning process models from example scenarios and from real learning traces. These methods are based on the well-known and successful concepts of process mining and workflow mining [10,11,12].

As in the area of business processes, there are learning processes with more flexibility and with less flexibility. Too much flexibility leads to deficient structure, with the problems mentioned initially. Too little flexibility is not realistic and might frustrate learners. We concentrate on structured learning processes with limited flexibility that are supported by appropriate supporting systems, just like workflows supported by workflow systems are not those business processes with maximal flexibility.

The paper is structured as follows. In the next section, we discuss learnflows and workflows and their similarities. Section 3 is devoted to suggestions for learnflow modeling with Petri nets. In Section 4 we argue that mining techniques can be applied to derive learnflows from runs of learning processes.

2 Learnflows and Workflows

In this section we investigate similarities and differences between workflow engineering, explored intensively since the 1990s, and the challenges of the newly established strand of educational modeling.

2.1 Comparison of the Workflow Reference Architecture and Learning Design Technologies

Business information systems support workflows that define the ordering of activities to be executed. In traditional information systems these processes are integrated in the software, i.e. hard-wired. They are thus not clearly visible and can only be modified with substantial effort. Modern ERP systems make the supported processes explicit and visible, yet the modification of processes (called customization) requires a major programming effort. More flexible in this respect are workflow systems which are a stringent application of the principles of business process engineering. Workflow systems are standard software in which the respective process logic is defined in a flexible way and according to the users' requirements. The resulting formal process model is an additional input for the system with a well-defined semantics. Its representation is interpreted by the system at run-time. Thus, process logic and functionality of the application are strictly separated and can be enhanced / evolved independently of each other.

The central component of a workflow system is the workflow engine. It controls and monitors the execution of a workflow by means of an explicit process model. It ensures that the tasks are correctly accomplished by the associated actors and subsystems. Besides the core workflow engine, several other components are needed for workflow management. Among these are tools to develop, define, and analyze processes, tools for interaction with users and applications, for administration, and for cooperation among multiple workflow systems.

The Workflow Management Coalition (WfMC) was founded in 1993 and is constituted today of more than 300 institutions representing all facets of workflow management, from vendors to users, and from academics to consultants (cf. *www.wfmc.org*). The WfMC takes care of standardization of concepts, terminology, and technology to promote interoperability and establishment of workflow technology in the market.

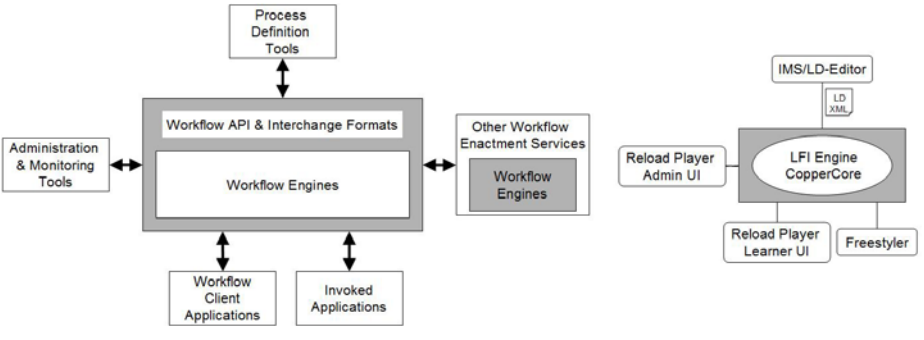

Fig. 1. Comparison of the Workflow reference architecture (left) and existing implementations of learnflow architectures (right)

Figure 1 shows on the left hand side the workflow reference architecture from the WfMC. It provides an overview of the main characteristics and components of a workflow system. We consider this reference architecture a suitable schema to compare workflow technology with the practice and technology in educational modeling and according systems.

The distinct components of the WfMC reference architecture have the following meaning:

Process Definition Tools: These include tools to edit and modify process definitions and tools for the analysis of these processes. The processes are imported into the workflow engine via a well-defined interface that takes the chosen representation format as input. A process definition specifies tasks, ordering of tasks and required resources (users / actors, external applications, etc.). Tasks can be executable concurrently or alternatively.

Workflow Engines: The workflow enactment service creates a process instance for each incoming case. Process instances are controlled by one or more workflow

engines. For each case the tasks to be executed are called work items. According to the process definition, a work item gets assigned available resources.

Workflow Client Applications: User interfaces allow users to interact with the workflow system, especially by means of worklists that show the currently active work items for a user.

Invoked Applications: This component consists of technical interfaces to external applications which can process, either fully automated or interactively, specific work items.

Other Workflow Enactment Services: The protocols and the functionalities are needed for interoperability between – and cooperative use of – different workflow systems.

Administration and Monitoring: Administrators and responsible persons of processes or of specific process instances use this user interface to manipulate and configure process parameters, inspect status information, and gain data for the analysis and re-design of processes.

In the field of educational modeling, a similar distinction of components is visible, yet an agreement on a similar standard architecture has not been reached. The formal, mostly XML-based, representations such as IMS/LD [13], LDL [14], PALO [15] or MoCoLADe [16] are usually not edited directly at XML-level but rather with specific editors on a more abstract level. Some of the representations are mainly visual, some of them are tree-structured textual representations. The models are interpreted by so called learning design engines. The engine controls – depending on the concrete approach – web-based user interfaces for the learners (e.g. the web player of IMS/LD) or external learning applications. An example of the latter is the remote control approach [17] which uses the collaborative learning application FreeStyler.

Some of the approaches provide a complete implementation of editor, engine, and learning environment, while other approaches such as the Collage tool or the MoCoLaDe modeling language use a semantic mapping of the created models to IMS/LD as a target language, i.e. they use IMS/LD as an educational assembler.

To administrate and observe the learning processes configuration and monitoring tools can be used, yet currently only basic functionality and support is available.

There is no generally agreed reference model for educational modeling yet. In analogy to the WfMC reference model, the right hand side of Figure 1 presents principles and interactions of functionalities / components of process based learning systems by means of existing software / practice in the learnflow field. There are obvious similarities with the workflow architecture. This analogy supports a potential for take-up of methods and concepts from the workflow to the learnflow field.

Research groups in educational design that also have a background in CSCW (computer supported cooperative work / groupware) have recently proposed to follow service oriented approaches [18] for architecture and components. For example, the GridCole system [19] integrated the freely available Open Source

IMS/LD-engine CopperCore (cf. *www.coppercore.org*) with tools that are available as Grid-services. Approaches stressing interoperability with workflow technology, such as BPEL, have also been brought up in educational modeling [20], but they represent still isolated research initiatives.

2.2 Specifics of Learnflows

Despite of the conceptual similarities between workflows and learnflows, one can find considerable discrepancies. There are several specifics of learnflows which restrict an immediate application of existing techniques from business process engineering. In particular, the following specifics have to be regarded by appropriate adaptations and extensions.

- In business processes, so called roles of actors determine which actors are allowed to execute certain tasks [21]. This approach stems from the concept of role-based access control (RBAC) known from IT-security [22]. The concept of roles in business processes is mainly based on individual responsibility and ability for a set of tasks, which is static after an initial assignment of roles to actors. Dynamic constraints on the task execution, e.g. separation of duties and binding of duties [23,22], and special rules for an appropriate allocation of actors to tasks have been considered [23,21] on top of the static role assignments. In contrary to this rigid role concept, the usage of roles in learning processes is frequently guided by exercising specific skills. Roles are dynamically changed and acquired during a learning process (e.g. using rotating roles [4]). The changes typically depend on the learning activities accomplished by an actor. Therefore, learnflows need an extension of static role concepts to dynamic ones that can take into account the learning history.
- For business processes, the focus is on an efficient accomplishment and completion of the process and its associated tasks. The generation of a product with guaranteed quality criteria or, more generally, the achievement of a business goal is important, while the participation of individual actors is only a minor concern [7,21]. In contrast, for learning processes the priority is that the learners involved in the tasks gain knowledge and experience. Neither a product nor an efficient completion of the process is important. Consequently, the focus of learning processes is on the learners and their learning success. This requires thorough and explicit modeling of the individual actors.
- Each activity might require group work and discussion, i.e. collaboration. The same holds for the entire learning process. These group activities usually have a high importance for the learning experience. Thus, the flexible and explicit representation of groups and - if needed - the dynamic re-arrangement of groups is a requirement for learnflow modeling.
- Finally, the phenomenon of concurrency, which allows an independent and simultaneous processing of tasks in business processes has to be considered with particular care in learning processes. Due to the distinct focus of attention, usually a single learner cannot accomplish tasks concurrently.

3 Modeling Learnflows with Petri Nets

In this section we present two modeling languages for learning processes that take specifically into account the requirements identified in the last subsection. To make use of expertise and experience from business process modeling, we build the proposals upon sound existing approaches from this field [7,24,21]. Besides the intended applicability for learning and teaching processes, we consider our approaches also useful for business processes with dynamic roles or with cooperative and collaborative tasks.

Both business processes and learning processes are usually modeled and communicated via diagrams. Petri nets [25] are a very popular formalism to model and analyze systems and processes involving concurrency. In particular, Petri nets and related modeling languages such as activity diagrams, event-driven process chains and BPMN are the standard graphical modeling languages for business processes [7,24,9]. Petri nets offer an intuitive syntax and a clear semantics as well as a large repertoire of analysis, simulation and synthesis procedures together with respective tools. While place/transition-nets (p/t-nets) are well suited for modeling the control flow of processes, high-level nets offer additional modeling features to represent data and time aspects. Business processes are often modeled by so called workflow Petri nets [7] which are a special kind of p/t-nets having distinguished initial and final places. In this section we suggest to use workflow nets also for the modeling of learnflows.

In the field of business process modeling, the transitions of a workflow net represent the tasks of a business process. The places model causal dependencies between the tasks. In order to model resources and resource allocation, usually roles are assigned to the transitions of workflow nets. It is assumed that conversely each actor possesses certain roles. An actor is only allowed to execute a task if he possesses the role associated to the task.

In the previous section we have argued that such a static role concept is problematic in the context of learnflows. To account for this difficulty, we introduce a more flexible role concept for learnflows in this section. We extend the established workflow modeling concepts by mechanisms to allow the change of actor roles while performing an activity. Because of the significance of the individual learners and their learning experience, we explicitly model the dynamic roles of actors by state diagrams. To also consider learning groups and collaboration, we further introduce collaborative tasks which are performed by several actors jointly. Since Petri nets allow an intuitive representation of the interplay between concurrency and non-determinism, a differentiated consideration of concurrency is enabled by the choice of Petri nets for modeling learning processes.

There exist a wide spectrum of Petri net languages that differ w.r.t. expressiveness of the single elements, understandability and notational effort. Generally, every Petri net can be represented as a particular high-level Petri net, e.g. using the terminology of colored Petri nets [26]. However, for different purposes and in different domains, variants and dialects have proven to be more appropriate because models are better understandable for users when the respective language is specifically tailored to a particular application area. As usual in computer

science, the semantics of such a domain specific language is given by translation to the more general language. Behavioral concepts and analysis methods and tools defined in the general setting thus apply to the specific one as well.

We will define two specific Petri net languages for learning processes and sketch the respective mappings to high-level Petri nets to give them the semantical foundation.

3.1 An Example

As an example we consider the following computer supported learning setting. Groups of three students use the tool FreeStyler [17] (*www.collide.info*) to learn the effect of different factors such as lightning conditions and CO2-concentration on the growth of plants. For this purpose, FreeStyler [17] provides several tabs with different functionalities (see Figure 2). There are, for instance, tabs to formulate questions, to create simple models or to import data from a simulation tool. In this example, the set of tabs corresponds to the set of supported learning tasks. Namely, we consider the following tasks:

- **In**troduction: The students read a short textual introduction to the problem.
- **Qu**estion: The students formulate the research questions that should be answered in experiments later on. They are guided by explanation prompts to step by step fill respective text fields.
- **Pl**anning: The students sketch a coarse model representing the influence of the different factors on the growth of plants. They relate nodes labeled with the different factors on a drawing area.
- **Mo**deling: On another drawing area, the students first refine the previous model (from Planning) by arc labels quantifying the effect of the factors. Then they test their model in a simple simulation.
- **Hy**pothesize: The students draw their hypotheses about the relationship of a factor and the growth of plants as a function in a coordinate system.
- **E**xperiment **1**: The students are directed to a simulation tool (e.g. BioBLAST). With the tool they can perform several experiments.
- **E**xperiment **2**: The students are directed to a second simulation tool.
- **Da**ta: The students investigate a table with precast results of lab experiments.
- **An**alysis: The students import their experimental data to a coordinate system. The resulting functions are analyzed and compared with the hypotheses.
- **Pr**esentation: The students summarize their results in a presentation. For this purpose a collection of text fields together with explanation prompts is provided.

In this context, a learnflow first describes the order in which the tabs have to be processed by the learners. Second, it determines the possible allocation of actors to learning tasks. It has to be specified whether a task requires a certain kind of collaboration and who is allowed to work on a tab where the latter depends on the learning history of the learners within the learning group.

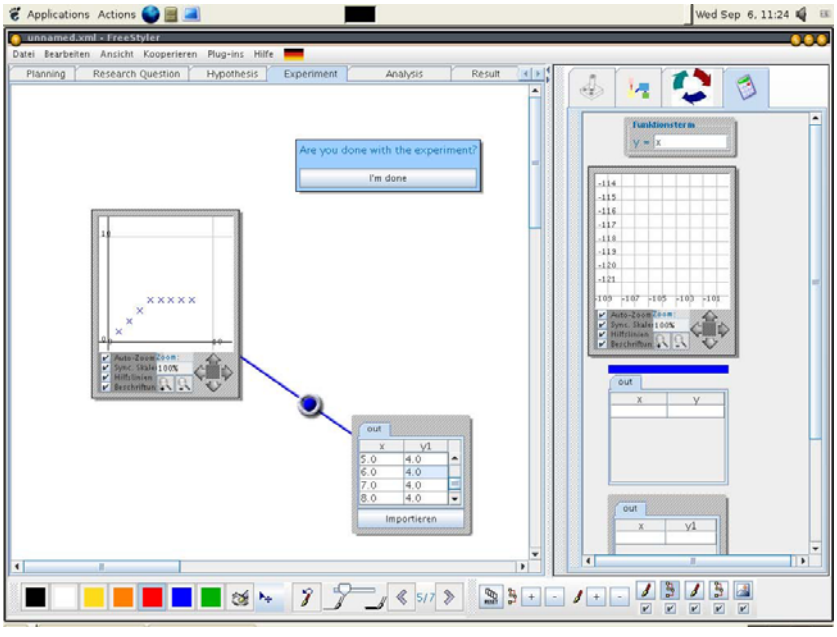

Fig. 2. Tabs in FreeStyler

The following example learning process describes groups of three learners working with FreeStyler. All three learners start with In. As soon as they have read the introduction, the two tasks Pl and Qu are executed independently. Qu requires one of the three learners. The learners have the freedom to choose who is assigned to the task Qu. In parallel to Qu, the two other learners jointly perform the task Pl. After Pl these two learners can optionally perform the task Mo which is also concurrent to Qu. When Qu, Pl and possibly Mo are accomplished, all three learners collaboratively work on Hy. Then, each of the three learners accomplishes one of the three independent tasks E1, E2 and Da. The learners can freely share these tasks among each other. When all three tasks are completed, the learners, again collaboratively, execute the task An. Subsequently, the task Pr concludes the learning process. Pr has to be performed by the learner that has initially executed the task Qu together with one of the other two learners.

3.2 Learnflow Nets

Learnflow nets present a first approach to extend the standard concepts of modeling workflows by Petri nets to learnflows. We model the control flow of a learnflow by a plain workflow net and depict actors and roles on top of the Petri net model by annotations. As in the case of workflows, we assume that there is a pool of actors possessing roles. Role annotations of transitions determine which actors are allowed to execute which tasks. However, the concept of role annotations is extended in two directions. First, to regard collaboration, instead

of a single role it is allowed to specify a multiset of roles meaning that actors possessing the roles given by the multiset have to execute the task jointly. Second, for more flexibility, it is allowed to specify alternative multisets of roles meaning that different role combinations are possible for executing the task. Besides the expressivity of role annotations, we also extend the relation of actors to roles in order to model dynamic role assignments. A single actor does not anymore just statically possess a role. Instead, he has now associated a state diagram modeling the dynamics of role possession. Each state of such a state diagram represents a role. Initially, an actor possesses the role given by the initial state of the state diagram. State transitions of the state diagrams, i.e. role changes of the respective actor, are labeled by transition names of the control flow Petri net. They are triggered when the actor executes a respective task, i.e. the transitions of the state diagram are synchronized with the transitions of the net.

Syntactically, we define a learnflow net by using the definitions of workflow nets and deterministic automata. We use the following notations. $\mathbb{N}$ denotes the non-negative integers. Given a finite set A, 2^A denotes the power set of A. $\mathbb{N}^A$ denotes the set of multisets over A. For $m \in \mathbb{N}^A$ we write $m = \sum_{a \in A} m(a) \cdot a$. By $|m| = \sum_{a \in A} m(a)$ we denote the cardinality of m. The set of all words (strings) over A is denoted by A^*. This includes the empty word λ.

Definition 1. *A workflow net is a tuple $N = (P, T, F, i, f)$, where*

- *P and T are disjoint finite sets of places and transitions,*
- *$F \subseteq (P \times T) \cup (T \times P)$ is a flow relation,*
- *$i, f \in P$ are an initial and a final place fulfilling $(T \times \{i\}) \cap F = \emptyset$ and $(\{f\} \times T) \cap F = \emptyset$, and*
- *for any node $n \in P \cup T$ there exists a directed path from i to n and a directed path from n to f.*

Definition 2. *A deterministic finite automaton over the finite set of input symbols T is a tuple $M = (Q, T, \delta, q_0)$, where*

- *Q is a finite set of states,*
- *$\delta : Q \times T \to Q$ is a transition function, and*
- *$q_0 \in Q$ is an initial state.*

Definition 3. *A learnflow net is a tuple $LF = (N, R, l, S, s)$, where*

- *$N = (P, T, F, i, f)$ is a workflow net,*
- *R is a finite set of roles,*
- *$l : T \to 2^{(\mathbb{N}^R)}$ is a labeling function assigning multisets of roles to each transition,*
- *S is a finite set of deterministic finite automata over the set of input symbols T such that the sets of states of two distinct automata are disjoint and such that the union of all state sets is R (S represents a set of role diagrams), and*
- *$s \in \mathbb{N}^S$ is a multiset of role diagrams representing the learners of the learnflow.*

As usual for workflow nets, initially we have one token in the distinguished initial place. The initial state of an automaton representing the initial role of an actor can vary, depending on the actual role (learner/teacher) or on the learning states of a learner obtained so far (novice/experienced/...).

Figure 3 together with Figure 4 show a learnflow net modeling our example learning process. Figure 3 illustrates the control flow aspect in the form of a workflow net with transition annotations. The annotations refer to the roles required for a task. The state diagram of Figure 4 complements the Petri net model. It represents a role diagram, which models the dynamic roles of the three learners in the pool of actors. Each learner corresponds to one instance of the state diagram.

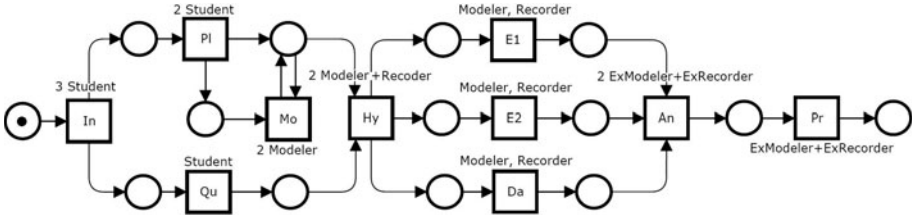

Fig. 3. Workflow net with role annotations

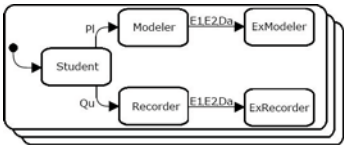

Fig. 4. State diagram representing dynamic roles

The dynamics of the learnflow net is as follows. The net gets an associated pool of actors. These actors cannot participate in other processes while the actual process is running. A task of the net can only be accomplished if this is possible in the current marking of the Petri net and if the pool of actors contains learners having the roles annotated at the transition. For instance, the collaborative task Pl requires two actors with the role Student. The roles of the learners are given by the role diagrams. In the example, the three learners may have the role Student, Modeler, ExModeler, Recorder and ExRecorder. Student is the initial role. The occurrence of a transition in the net can change the roles of the involved actors. Such a change is modeled in the role diagram by a state transition with the task name as the input symbol. Therefore, in our example the task Pl causes the two students executing Pl to switch to the role Modeler, which is later on required to perform the task Mo. When the net has received its final state (one token in the final place) the actors can take part in other learning processes, including further instances of the one just considered.

For simplicity of the role diagrams, we use the following convention. We omit transitions from a state to itself, i.e. if an actor performs a task which does not explicitly cause a change of his role in the state diagram, he stays with his role. For instance, the task Mo does not change the role of an actor with the role Modeler and therefore can be neglected in the state diagram.

While the Petri net from Figure 3 exactly represents the control flow of our example learning process, the annotations together with the associated state diagram of Figure 4 model the specified rules for the distribution of the learning tasks. The crucial aspect are the role changes. They allow to represent the given restrictions as well as the specified degree of freedom for the distribution of the tasks.

For instance, any two learners can execute the task Pl, since they all have the role Student at the beginning. The resulting role change from Student to Modeler then has two immediate effects. On the one hand the two learners are afterwards allowed to execute the task Mo which requires two Modelers. On the other hand the two learners cannot anymore execute the task Qu, since they do not anymore possess the default role Student which is required for this task.

We have a similar situation with the three tasks E1, E2 and Da. Each of these tasks can be executed by a Modeler or a Recorder. At this stage of the process, each learner has one of these two roles and can therefore be assigned to the tasks. However, the role changes associated with these tasks ensure that each learner can only accomplish one of the three tasks. A Recorder becomes an ExRecorder and a Modeler becomes an ExModeler. Finally, as desired, the role history guarantees that the learner who has formulated the question has to participate in the presentation, since he is the only learner possessing the role ExRecorder, i.e. this role stores the information that the learner has performed the task Qu. Additionally, the task Pr requires one of the two other learners possessing the role ExModeler.

The example shows that learnflow nets allow to appropriately represent collaborative activities as well as dynamic roles and the learning progress of learners within a learning process. Nevertheless, the modeling language is simple and intuitive. It naturally extends well established modeling approaches from the domain of business process management. As in the case of business process modeling, the approach supports a clear separation of the control flow perspective and the role perspective of a learnflow, but still regards both perspectives in one model.

3.3 Expressiveness of Learnflow Nets

The central new feature of learnflow nets is the concept of dynamic roles. The learning progress and the learning history of actors are encoded in role diagrams. This is an intuitive approach since, after performing a learning task, an actor often has additional knowledge, skills or responsibilities. Therefore, determined by his new status, an actor has a new role in the learning process. This role can then in a natural way be used to constrain the allocation of later tasks. This concept to represent the distribution of learning tasks allows to model in detail which tasks have to be done by certain actors and in which cases

the actors can freely decide about task assignments. Using static roles, it is in many cases not possible to allow a degree of freedom for the learners. Omitting roles is also problematic, since then one learner would be allowed to execute all tasks. Therefore, without dynamic roles additional constraint specification languages, as e.g. introduced in [23,22], are necessary for an appropriate modeling of learnflows. However, we consider this more complicated and less intuitive than using dynamic roles for learners.

Formally, the behavior of learnflow nets can be defined by a translation into a special class of high-level Petri nets, namely colored Petri nets [26]. Colored Petri nets have a well-established occurrence rule which can then be applied for learnflow nets. We here do not formally define the translation. Instead we explain the ideas and illustrate them by our example learnflow net. The systematic formal translation can easily be deduced from this example.

All the standard Petri net components of the learnflow net are kept in the high-level Petri net model. The annotations of the transitions and the state diagrams are translated into one high-level place modeling the pool of resources resp. actors. The color set of this place is given by the possible roles of learners. The initial marking contains, for each instance of a state diagram, one token of the kind given by the initial state of the state diagram. The place has an outgoing and an ingoing arc connected with each transition of the net. A transition consumes tokens from the resource place as given by its annotation. For each consumed token, there are two cases. Either the state corresponding to the token type in the state diagram enables a state transfer labeled with the name of the considered transition or it does not enable such a transfer. In the first case, the transition produces a token of the kind given by the follower state of the state diagram in the resource pool. In the second case, it produces a token of the same kind as the consumed token. For classical workflow models with static role annotations, an analogous translation is possible, but in this situation the first case never occurs, i.e. each transition produces the same tokens in the resource pool that the transition consumes from the pool. Alternative role annotations of learnflow nets can be formulated by means of respective expressions of the high-level transitions.

Figure 5 illustrates the described translation for our example learnflow net. The resulting high-level Petri net does not any more show explicitly the dynamic roles and the behavior of the learners. Moreover, it is quite difficult to read and understand the high-level Petri net. Therefore, for modeling purposes, the original representation should be preferred. The high-level view is used for formal considerations.

3.4 Actor Learnflow Nets

In our second modeling approach, called actor learnflow nets, we go one step further. We remove the pool of actors and embed the role diagrams as tokens into the Petri net, representing the control flow of the process model. In this way the progress of actors within a learnflow is represented by the location of respective tokens in the Petri net. This approach is inspired by the idea of

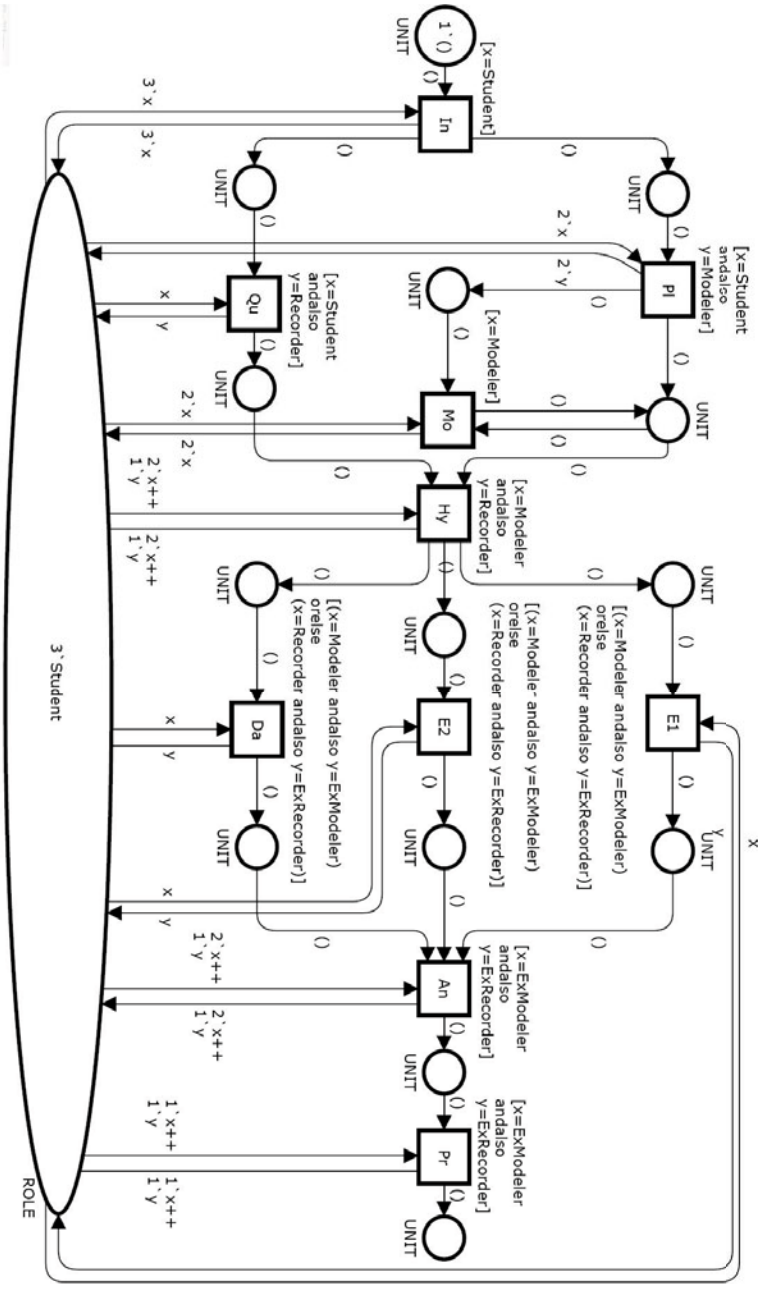

Fig. 5. Translation of the example learnflow net into a colored Petri net (in CPN tools http://cpntools.org/)

modeling with nets-in-nets and in particular by the papers about modeling of multi agent systems, inter-organizational workflows and adaptive workflows with object nets (e.g. [27]). However, our modeling concept is a lot more restrictive and simple than object nets [28].

In contrast to learnflow nets an actor learnflow net is not anymore based on a workflow net. It is a Petri net consisting of two kinds of places, "plain" Petri net places which can be marked by black tokens and role places which can be marked by role diagrams. The plain places can be considered just like places of a usual workflow net modeling control flow. The role places represent the locations of actors. They are connected to transitions by arcs having arc weights. For simplicity, we do not allow high-level arc expressions as allowed in object nets. Besides representing actors, the role places also encode control flow aspects. They are only supplemented by plain places if this is necessary for expressing advanced control flow structures.

An actor learnflow net has an initial marking assigning multisets of role diagrams to role places. The role diagrams represent the learners involved in the learning process. They are basically treated just like black tokens, i.e. they are not distinguished. Actor learnflow nets include a crucial restriction. For each transition the number of role diagrams consumed by the transition, i.e. the sum of the ingoing arc weights, has to equal the number produced by the transition, i.e. the sum of the outgoing arc weights. Since each role diagram represents a learner participating in the process, this preservation rule ensures that learners are not created or deleted, instead they are just moved forward in the learnflow. When firing a transition, it produces the same role diagrams that were consumed by the transition. Since the arcs are annotated only by the numbers of role diagrams to be produced or consumed, only the numbers of role diagrams consumed from each place and forwarded to each place are specified. This way, the modeling language is kept comparably simple whereas the abstraction from individual identities does not harm.

An important aspect of actor learnflow nets is that they allow transition annotations modeling restrictions on the role diagrams consumed by a transition. These annotations have the same form and the same meaning as in the case of learnflow nets. The annotations specify alternative multisets of roles. The states of the role diagrams consumed overall by a transition have to match one of these combinations of roles. Therefore, the annotations again model which role combinations are allowed for executing a task. Remark that we have to require that the cardinality of the annotated multisets equals the number of role diagrams consumed by the transition. Concerning the role diagrams themselves, analogously as in the case of learnflow nets, their transitions are synchronized with the transitions of the Petri net. They cannot spontaneously change their states as in the case of object nets. The syntax of actor learnflow nets is as follows.

Definition 4. *An actor net is a tuple* $N = (P, P', T, W, F)$*, where*

- P*,* P' *and* T *are pairwise disjoint finite sets of role places, plain places and transitions,*

- $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ *fulfilling* $\sum_{p \in P} W(p,t) = \sum_{p \in P} W(t,p)$ *is a flow relation (specifying arc weights) for the role places and*
- $F \subseteq (P' \times T) \cup (T \times P')$ *is a flow relation for the plain places.*

Definition 5. *An actor learnflow net is a tuple* $ALF = (N, R, l, S, s)$*, where*

- $N = (P, P', T, W, F)$ *is an actor net,*
- R *is a finite set of roles,*
- $l : T \to \{X \subseteq \mathbb{N}^R \mid \forall x \in X : |x| = \sum_{p \in P} W(p,t)\}$ *is a labeling function assigning roles to each transition,*
- S *is a finite set of deterministic finite automata over the set of input symbols* T *such that the sets of states of two distinct automata are disjoint and such that the union of all state sets is* R *(*S *represents a set of role diagrams) and*
- $s : P \to \mathbb{N}^S$ *is an initial marking assigning multisets of role diagrams representing the learners of the learnflow to the role places.*

We assume that in the initial marking plain places are unmarked.

Figure 6 depicts an actor learnflow net representing our example learning process. In addition to role places, the net includes only one (small) plain Petri net place for representing additional control flow aspects. The three learners of the learning process are all represented by an instance of the same role diagram representing the dynamic roles of the learners. In the initial marking they are located in the same role place. The transitions are annotated by labels referring to the roles required for a task.

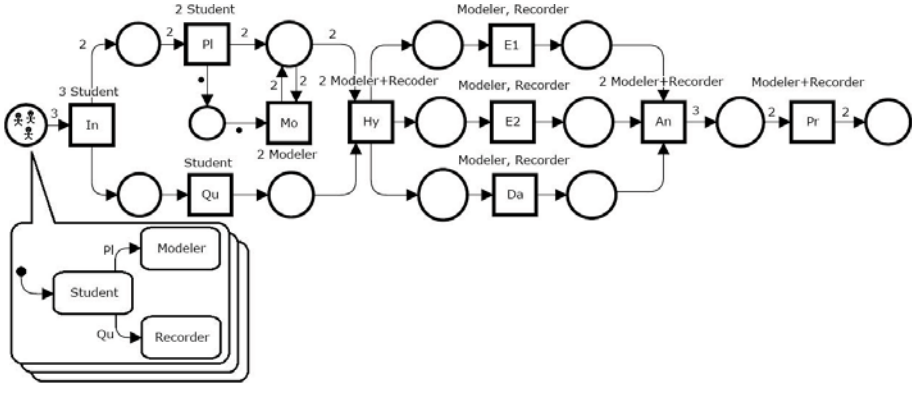

Fig. 6. Actor learnflow net

The dynamics of an actor learnflow net is as follows. A transition of the net can be fired if this is possible in the Petri net when treating role diagrams as black tokens and if the role diagrams consumed overall by the transition can be chosen in accordance to the roles annotated at the transition. These role diagrams are then distributed to the places in the postset of the transition. The distribution is non-deterministic, but it has to regard the arc weights. Moreover,

the occurrence may change the states of the role diagrams analogously as in the case of learnflow nets.

In our example, we consider the transition Hy. An occurrence of this transition requires three learners, namely two Modelers and one Recorder. Two learners have to be in the upper place of the preset of the transition and one learner in the lower place. It does not matter whether the Recorder is in the upper or the lower place. Since, according to the role diagrams, neither a Recorder nor a Modeler changes its role when executing Hy, the three state diagrams are not modified and forwarded to the three postset places. One of the three role diagrams is added to each of these places. It does not matter to which place the Recorder is forwarded.

In this approach the distribution of the learning tasks is not anymore just represented by the dynamic roles. Instead, also the location of the role diagrams is important. For instance, as explained, the transition Hy distributes the three role diagrams to the three places enabling E1, E2 and Da. This ensures that each of the learners has to execute one of these tasks. Similarly, the transition In distributes two learners to the task Pl and one learner to the task Qu.

In contrast to the learnflow net from the last subsection, it is noticeable that the transitions E1, E2 and Da do not anymore trigger a role change. This is because the distribution of the learners to the three tasks is given by their location in the net and not anymore by a role change. That means, actor learnflow nets represent progress aspects explicitly by the location of role diagrams in the Petri net. As a consequence, role diagrams become simpler.

The example shows that actor learnflow nets are appropriate for comprehensibly modeling learning processes. The process perspective and the role perspective are embedded into one model. A nets-in-nets approach is applied, where tokens are role diagrams representing actors. Thereby, the role of an actor is given by the current state and the progress of the actor within the learning process is given by the location of the diagram.

We again show, using our example net, that the behavior of actor learnflow nets can formally be defined by a translation into colored Petri nets. The transitions and plain places are kept and the role places are translated into high-level places which have a color set given by the possible roles of learners. The initial marking is given by the initial roles of the state diagrams. The different nondeterministic possibilities of consuming and producing role diagrams by transitions are translated into respective firing modes of transitions. This is realized by assigning variables to respective arcs and regarding the allowed assignments of the variables by means of transition guards. These guards in particular have to regard the dynamics of roles given by the role diagrams.

Figure 7 shows the described translation by our example actor learnflow net. As in the case of learnflow nets, the colored Petri net associated to an actor learnflow net hides the dynamics of roles and is difficult to understand. Consequently, it is only useful for a formal analysis of actor learnflow nets.

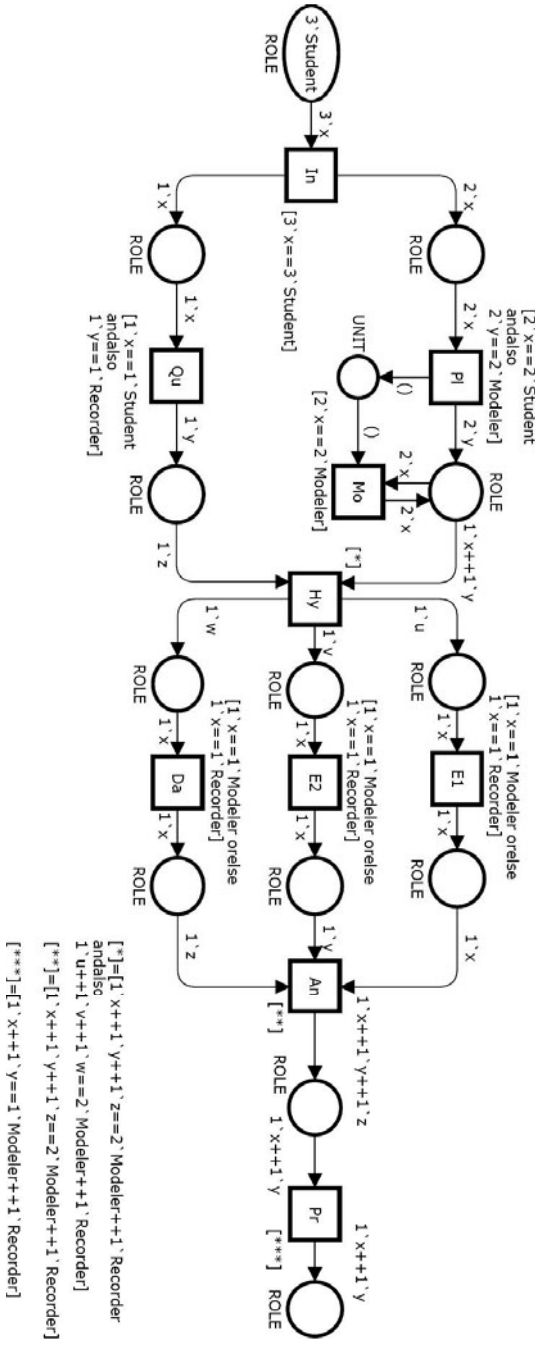

Fig. 7. Translation of the example actor learnflow net into a colored Petri net (in CPN tools http://cpntools.org/)

3.5 Discussion

Concluding, both learnflow nets and actor learnflow nets model the control flow of learning processes in a formal as well as illustrative way by means of Petri nets. They allow an appropriate representation of dynamic roles, collaboration and learning progress. The approaches differ w.r.t. their semantical clearness where none of the approaches outmatches the other one.

Learnflow nets clearly distinguish the modeling of the control flow and the resource perspective, thereby extending well-known concepts for workflow modeling. Due to the use of plain black tokens, the model is simple and allows well established formal analysis methods.

In contrast, actor learnflow nets use complex tokens represented by role diagrams having a current state. As a consequence, analysis of an actor learnflow net is more difficult and requires tailoring of known techniques. Also, the readability of the graphical representations might become worse. Finally, the role perspective is embedded into the control flow such that the clear separation known from workflow approaches is lost. On the other hand, by this approach the learning progress of each actor is explicitly represented in the control flow model. Therefore, progress dependant role changes as in the case of learnflow nets are not any more necessary. Thus, role diagrams have a more limited clear semantics and become simpler. This phenomenon becomes obvious in our example when for instance regarding the task E1. This task initiates a learning progress of the involved learner. In the learnflow net the progress is represented by a change of the role of the learner. The new role then does not allow an execution of E2 and Da by this learner anymore. In the actor learnflow net such progress-dependent role change is avoided. It represents the progress of the learner in a more elegant way by moving the actor token forward in the Petri net model.

Lastly, both modeling approaches focus on the representation of dynamic roles and regard group learning by means of collaborative activities. Thereby, a learning group is usually given by one process instance and the dynamics within a learning group is only implicitly considered. In view of representing re-arrangements of groups and an explicit modeling of group dynamics, we suggest two natural and useful extensions of learnflow nets resp. actor learnflow nets. First of all, when modeling groups by process instances, an extension of the modeling languages by global (or static [29]) places which allow to model dependencies of process instances would be helpful to regard dynamic group re-arrangements. Second, groups and group dynamics can also be modeled analogously as roles, i.e. the role diagrams of the actors can additionally capture information about group membership of actors. However, in contrast to roles it is important to represent the overall dynamics of the groups (which actors belong to a group at a particular time?). This dynamics can only implicitly be observed by group-memberships of single actors. Therefore, an extension of the modeling approach which at any time explicitly represents the learning groups is promising. This can for instance be achieved by a respective grouping of the state diagrams representing the learners.

4 Learnflow Mining

After having adapted modeling languages for business processes to learnflows, we in this section transfer the modeling and analysis technique of process mining [10,11,30,12]. The creation of a business process model can be automated or at least supported when example executions of the process are known. For this purpose, certain protocol files, so called event logs, which record the tasks executed within an information system can be used. Methods to automatically generate process models from event logs are known as process mining or workflow mining. They have recently gained a lot of attention in research and have also found their way into industrial tools [10,30,31]. We suggest to use process mining methods and tools for the generation of learnflow models in a similar way as in the case of business processes.

Teachers are not familiar with explicit modeling of learning processes. Usually, lesson plans are restricted to sequential considerations in table form. These tables cannot appropriately represent complex learnflows such as spiral approaches or learnflows including dynamic learning groups working concurrently. Moreover, the explicit definition of learning scenarios by a teacher often suffers from the so called expert blindspot [32], i.e. the teacher only considers the obvious scenarios and does not anticipate unconventional approaches. Both aspects show that real learning scenarios should be used for generation of learnflow models. We call the automatic generation of learnflow models from example scenarios learnflow mining, in analogy to workflow mining. We initiated the discussion about the application of process mining in the area of learning processes and coined the term learnflow mining in [33]. Independent work using educational mining approaches have been used to identify decision making behavior in virtual teams [34] and to identify behavior of students solving multiple choice tests [35]. While the latter uses single learner data, thus focusing on individual aspects, the first is also located in the field of collaborative learning. Yet, it concentrates on the activities and not on social or learning roles, so that both studies are similar to our work w.r.t. control flow mining while role mining is not tackled in [34,35]. Related work concerning role mining is organizational mining [12], which is however limited to static roles and organizational units.

Learnflow mining can help a teacher to construct valid lesson plans using advanced modeling languages. In particular, due to automation the effort for generating models can be reduced. For many teachers, this might be a crucial requirement for applying learnflow models. In general, analogously to experiences from the area of cognitive tutors [36] where example representations of general rules are applied, focusing on example learning scenarios instead of whole learnflows is more intuitive and less complex for teachers.

For the generation of example learning scenarios we suggest two approaches (which can also be combined), both requiring that the events of a learning system are recorded in log files.

- The teacher generates learning scenarios by demonstration on the learning system. He executes the scenarios with the learning system, thereby

recording them in an event log. Then he possibly categorizes the scenarios of the log w.r.t. correctness or their potential for learning success.
– Real example scenarios of learners using the learning system are logged. The learners can freely work with the learning system such that log files are generated. For a classification, either a self-assessment of the learners in the sense of collaborative filtering can be applied or the teacher can evaluate the learning results by means of tests.

In both cases the result is a classified event log. The classification is useful to filter the relevant scenarios. The filtered event log can be used as input for process mining algorithms in order to generate a learnflow model which represents the real learning scenarios. In the following we discuss this concept in detail for learnflow nets.

The central aim of mining a process model from its specified behavior is to create a model that has exactly the behavior specified, no matter if we consider workflows or learnflows. However, we can only expect such a result if the specified behavior contains all and only legal runs. This assumption is unrealistic for larger process models because the number of legal runs can grow exponentially with the size of the model. Moreover, when starting with observed behavior, it is not easy to say whether a log is complete or whether some legal runs just never appear in the log. The situation becomes even more complicated if a log contains runs that are not legal (so called noise) or if a log contains runs that are successful, i.e., lead to a positive end, and runs that are not successful but still legal. The latter is particularly relevant for learnflows because a legal learning process can be successful or unsuccessful.

Therefore, in general it is neither possible nor desirable to gain process models that precisely have the specified behavior. The best we can expect is a model that has at least the specified behavior, is reasonably simple and has not too much additional behavior. Moreover, the model should be sound, i.e., it has no superfluous elements, does not run into deadlocks and ends with a proper end state. Roughly speaking, these are the main goals in process mining. We cannot expect to gain more than that when mining learning processes.

Usually, the generated (learning) process models are the starting point for a manual revision. At this stage, relevant properties of the model are analyzed and the model might be modified so that it enjoys desirable properties. The specified behavior is compared with the actual behavior to check whether this model is really suitable as a basis for legal processes in computer supported learning environments.

4.1 Control Flow

The methods to mine the control flow of a learning process are directly deduced from process discovery approaches known from process mining. The basic ideas of these approaches are as follows [10,11]. Every process-aware information system supports a set of tasks of a corresponding operational process. The execution of a task is an event. Typically, an event is stored together with information

about the corresponding process, the process instance, the execution time, the involved actors and maybe some more information as for instance associated data elements. Within a process instance of a given process, the events can be ordered by their time stamps. Therefore, each process instance defines a sequence of events. The sequences of events associated to a certain process can then be used to identify the control flow of the process. The benefit of this approach is the following. While the manual construction of valid process models is a difficult, intricate and erroneous task, a process mining algorithm automatically generates a process model which matches the actual flow of work. The generated model can then be used for verification, analysis and optimization issues or even for controling the operational process by an engine.

For learnflow mining we formally consider an event log of one process as an ordered sequence of events where each event is a triple of the form (process instance, task, set of actors). We assume that the ordering is given by the time stamps of events. Moreover, we abstract from additional information such as data.

Definition 6. *Let T be a finite set of tasks, I be a finite set of process instances and A be a set of actors. An event is an element of $T \times I \times 2^A$. An event log is a sequence $\sigma = e_1 \dots e_n \in (T \times I \times 2^A)^*$.*

Given a log, the sequences of task occurrences given by single process instances are called control sequences.

Definition 7. *Let $\sigma = e_1 \dots e_n \in (T \times I \times 2^A)^*$ be an event log. Given a process instance $i \in I$, we define the function $c_i : T \times I \times 2^A \to T \cup \{\lambda\}$ by $c_i(t, i', U) = t$ if $i = i'$ and $c_i(t, i', U) = \lambda$ otherwise. Then $c_i(e_1) \dots c_i(e_n)$ is a control sequence of σ for the instance i.*

There are many workflow mining methods generating a Petri net model from the set of control sequences of a given event log such that the model represents the behavior given by the control sequences [10,11,30]. That means, the generated model represents the control flow of the underlying workflow. In our setting of learnflow mining, besides the control flow we also have to consider the dynamic role behavior. In the case of learnflow nets, the control flow perspective and the role models are clearly separated and can be mined one after the other. We here first consider the problem of mining the control flow given by an event log of a learning system. In the next subsection we discuss the mining of dynamic roles.

The control flow perspective of learnflow nets is given by standard workflow nets. To generate the workflow net underlying a learnflow net we can therefore apply any algorithm from workflow mining. We demonstrate this using our example learning setting. We assume that we have several learning groups processing the tabs offered by FreeStyler and thereby using an ordering and a distribution of the tasks as given by our example learning process. Each processing of a tab is recognized by FreeStyler and recorded in an event log. In this way, each learning group creates a process instance. Note that, as described before, the set of process instances can be filtered by appropriate criteria, and the teacher can also add certain desired process instances by demonstration. An extract of a respective event log is illustrated in Figure 8. We assume that the whole log (which is

large and cannot be shown here) is complete for our example in the sense that all possibilities to work on the learning tasks according to the example learning process are included in the log.

Process "Plants"		
Instance	Task	Actor
Group A	In	Chuck, Wes, Susan
Group B	In	Robin, Andy, George
Group C	In	Chris, Anne, Dave
Group C	Pl	Anne, Dave
Group C	Qu	Chris
Group A	Qu	Wes
Group C	Mo	Anne, Dave
Group B	Pl	Robin, George
...	...	...

Fig. 8. Events recorded by FreeStyler

In the shown part of the example log we have the following control sequences: $In, Qu, ...$ (Group A), $In, Pl, ...$ (Group B) and $In, Pl, Qu, Mo, ...$ (Group C). Already for our small example process, the integration of the control sequences to one Petri net model exhibiting the behavior given by the sequences is not trivial. Therefore, we use the event log as input of a process mining algorithm. The workflow mining tool ProM [30] offers a lot of different mining algorithms which automatically generate a Petri net model representing the behavior of the learners as given by the log. For our complete example log file of FreeStyler we applied a mining algorithm which is based on exact synthesis methods and region theory [11]. With this tool we were able to mine the workflow net from Figure 3 (without annotations).

Thus, it is possible to mine the control flow perspective of a learnflow net which can then be used for analysis purposes.

4.2 Roles

The role names do not occur in our example log. Therefore, we cannot expect to correctly mine the role names that appear in the learnflow model of Figure 3 and 4. Our aim is to mine role diagrams and transition annotations which define the same restrictions for the assignment of tasks as given in the example process. For this purpose, the behavior of the actors recorded in the log is important. Consequently, the initial point for mining the role perspective is given by the control sequences, where now each event is stored together with the involved actors. We call these sequences learning sequences.

Definition 8. *Let $\sigma = e_1 \dots e_n \in (T \times I \times 2^A)^*$ be an event log. Given a process instance $i \in I$, we define the function $l_i : T \times I \times 2^A \to (T \times 2^A) \cup \{\lambda\}$ by $l_i(t, i', U) = (t, U)$ if $i = i'$ and $l_i(t, i', U) = \lambda$ else. Then $l_i(e_1) \dots l_i(e_n)$ is a learning sequence of σ.*

In our example, Group C defines the learning sequence $(In, \{Chris, Anne, Dave\}), (Pl, \{Anne, Dave\}), (Qu, \{Chris\}), (Mo, \{Anne, Dave\}), ...$

For any learning sequence and each student occurring in the sequence, we consider the sequence of task occurrences (without information about students) the student is involved in. Such a projection of a learning sequence onto a single actor is called actor sequence.

Definition 9. *Let $(t_1, U_1) \dots (t_n, U_n)$ be a learning sequence of an event log $\sigma \in (T \times I \times 2^A)^*$. Given an actor $u \in A$, we define the function $a_u : T \times 2^A \to T \cup \{\lambda\}$ by $a_u(t, U) = t$ if $u \in U$ and $a_u(t, U) = \lambda$ otherwise. Then $a_u(t_1, U_1) \dots a_u(t_n, U_n)$ is an actor sequence of σ.*

In our example, Anne executes the actor sequence $In, Pl, Mo, ...$ in Group C.

It is possible that different actors have different role behavior, i.e. different associated role diagrams. In such a case we have to divide the actors of a process instance into respective classes. Such a behavior does not appear in our example. For simplicity, we assume that all actors have the same role behavior. For instance, in our example all actors are simply learners.

Now the crucial step is to integrate all actor sequences into a deterministic state diagram in the form of a tree. That means, the states of the diagram are determined by the history of previously performed tasks (also regarding the order of the events) and are named accordingly, i.e. the states are given by the prefixes of the actor sequences. This diagram can then be used as a role diagram. Formally, we apply a very simple technique for learning a deterministic finite automaton from a set of words [37]. More sophisticated constructions such as the one of Myhill-Nerode cannot be applied here since the possible role combinations for executing a task as given by the log have to be regarded (see the Simplification Rule at the end of the section).

Definition 10. *Let AS be the set of actor sequences of an event log $\sigma \in (T \times I \times 2^A)^*$. The deterministic finite automaton $RD = (Q, T, \delta, q_0)$ where*

- $Q = \{t_1 \dots t_i \mid t_1 \dots t_n \in AS, 0 \leq i \leq n\}$,
- *δ is defined by $\delta(t_1 \dots t_i, t_{i+1}) = t_1 \dots t_{i+1}$ if there exists $t_1 \dots t_n \in AS, 0 \leq i < n$ and $\delta(t_1 \dots t_i, t_{i+1}) = t_1 \dots t_i$ otherwise, and*
- $q_0 = \lambda$

is called role diagram of σ.

For our complete example log, the resulting role diagram is given in Figure 9.

The diagram encodes the complete history of each actor. In learnflow nets we assume that the role of an actor can only change in the case the actor executes a task. Therefore, when consistently defining annotations of transitions in the Petri net, it is possible to exactly represent the task allocation given in the log. To define the annotation of a certain transition we have to take the learning sequences into account. When a transition occurs in a learning sequence, the histories of the involved actors within the learning sequence define one possible role allocation. There might be different possible role combinations given

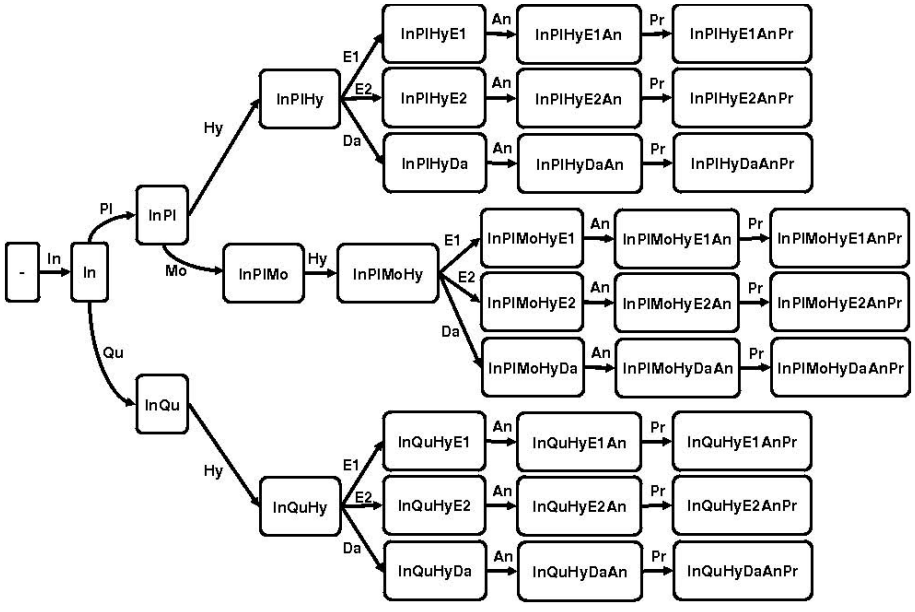

Fig. 9. Mined role diagram

by all the learning sequences. All these alternative role assignments define the annotation of the transition.

Definition 11. *Let* $(t_1, U_1) \dots (t_i, U_i) \dots (t_n, U_n)$ *be a learning sequence of an event log* $\sigma \in (T \times I \times 2^A)^*$ *and* $RD = (Q, T, \delta, q_0)$ *be the role diagram of* σ*. A possible allocation of roles for the task* t_i *is given by the multiset of roles* $\sum_{u \in U_i} a_u(t_1, U_1) \dots a_u(t_{i-1}, U_{i-1}) \in \mathbb{N}^Q$ *(*a_u *is defined as in Definition 9).*

The role annotation $l(t)$ *of a task* $t \in T$ *w.r.t.* σ *is the set of all possible allocations for* t*. The resulting function* $l : T \to 2^{(\mathbb{N}^Q)}$ *is called role labeling function of* σ*.*

In the example learning sequence of Group C, the task Mo is performed by Anne, who has done In and Pl before, together with Dave, who has the same learning history. Therefore, $2 \cdot InPl$ is one possible role allocation for Mo. Since Mo is always done by $2 \cdot InPl$, the role annotation $l(Mo)$ of Mo w.r.t. the given log is $\{2 \cdot InPl\}$.

Altogether, we are now able to mine a complete learnflow net from a log of a learning process. In the previous subsection we have shown how to generate a workflow net representing the control flow of the learning process. In this subsection we have shown how to mine a role diagram together with role annotations representing the resource perspective of the learning process. Combining these two models yields a learnflow net representing the behavior of the learning process. We refer to this net as the learnflow net mined from the log. The number

of role diagram instances of this learnflow net is given by the maximal number of actors occurring in a learning sequence of the log.

Definition 12. *Let $\sigma \in (T \times I \times 2^A)^*$ be an event log and $N = (P, T, F, i, f)$ be a workflow net mined from σ. Let further LS be the set of learning sequences of σ, $RD = (Q, T, \delta, q_0)$ be the role diagram of σ and $l : T \to 2^{(\mathbb{N}^Q)}$ be the role labeling function of σ.*

The learnflow net mined from σ w.r.t. N is defined as $LF = (N, Q, l, \{RD\}, k \cdot RD)$ where $k = max_{(t_1,U_1)\ldots(t_n,U_n)\in LS} |\bigcup_{i=1}^{n} U_i|$.

The examples given in this subsection illustrate that the possible distributions of tasks to actors allowed by the learnflow net mined from a log coincide with the distributions of tasks given in the log. Therefore, if the control flow has precisely been mined, the behavior of the mined learnflow net coincides with the behavior given in the log up to isomorphism and renaming of the roles. For instance, the behavior of the learnflow net mined from the example log is equivalent to the behavior given by the log and thus also equivalent to the behavior of our original learnflow net given in Figures 3 and 4.

In our example, the mined role model is by far larger than necessary. This is the case because the model represents the whole history of each learner. Therefore, an important issue is to simplify the mined role models by merging roles.

Given a mined role model, the central idea for simplification is to merge roles, i.e. states, having an equivalent future. Of course, when merging roles also the annotations in the corresponding Petri net model have to be renamed consistently. The behavior of a learnflow model is not changed by this simplification. For instance, all the leafs in our example role diagram in Figure 9 have no follower roles and can thus be merged. Also the previous nine roles have the same follower role triggered by the same task. However, we here have to consider the collaborative aspect. It is not allowed to merge two roles having the same future in the role diagram if switching the two role names changes the annotation of collaborative tasks in which both roles are involved. In our example, the task Pr is a collaborative task which is performed by two actors. Thereby, it is important that one of the actors is the one that executed Qu. Consequently, switching a role name including Qu and a role name not including Qu changes the annotation of Pr. Thus, only the upper six and the lower three roles can be merged. Altogether, we formulate the following rule:

Simplification Rule. *Let $RD = (Q, T, \delta, q_0)$ be a role diagram and $l : T \to 2^{(\mathbb{N}^Q)}$ be the role labeling function of an event log σ. Two states $q_1, q_2 \in Q$ can be merged if the following two conditions are satisfied:*

- $\forall\, t \in T : \delta(q_1, t) = \delta(q_2, t)$ *or* $(\delta(q_1, t) = q_1$ *and* $\delta(q_2, t) = q_2)$
- $\forall\, t \in T$: $(\exists\, x_1, x_2 \in l(t) : |x_1| > 1$ *and* $q_1 \in x_1, q_2 \in x_2) \implies l(t) = \{s(q_1, q_2, x) \mid x \in l(t)\}$, *where* $s : Q \times Q \times \mathbb{N}^Q \to \mathbb{N}^Q$ *is given by* $s(q_1, q_2, x) = \sum_{q \in Q \setminus \{q_1, q_2\}} x(q) \cdot q + x(q_1) \cdot q_2 + x(q_2) \cdot q_1$.

When no more roles can be merged according to this simplification rule, an additional technique can be applied. Many of the role transitions from a state to itself (which are not shown in the illustrations of role diagrams) are irrelevant. An irrelevant role transition cannot occur in the learnflow model since it is forbidden by the control flow. Therefore, changing such transitions does not affect the behavior of the learnflow net. But such changes can be used to align the future of two states in a role diagram in view of merging them.

Using this technique for our mined role diagram of Figure 9 reduces it to a diagram which is isomorphic to the original example role diagram of Figure 4. That means, with the simplification rule we are in this case able to recreate the original learning process model from a complete log file. Only the role names that are not present in the log files cannot be mined. They can be chosen arbitrarily by the teacher.

5 Conclusion

In this article we systematically carried over methods and techniques from workflow management to the modeling of learnflows. First, we worked out the similarities and differences of workflow and learning processes. In particular, we have compared relevant implementations in the two areas. As a basis for further considerations we then have developed and discussed two modeling languages for learnflows which are based on workflow modeling concepts, namely learnflow nets and actor learnflow nets. Finally, we have identified the field of process mining as particularly interesting for a concrete method and technology transfer. Under the name learnflow mining, we suggested to use mining methods for the generation of learnflow models from real learning scenarios. Using real scenarios and automatically generating more general learnflow models on the one hand disburdens the teacher when designing models and on the other hand enables the immediate usage of concrete learning experience (e.g. for supporting the learners within the learning process). We have discussed both, mining of the control flow of a learning process and mining of the role behavior of the learners within the learning process.

As in the area of workflow mining [38], for the development of learnflow mining tools a standardization of log formats of learning systems – which is already being discussed in the field of educational modeling – is an important aim. Moreover, mining tools have to support preprocessing of the event logs and postprocessing of the generated process models. For instance, an event log recording all mouse clicks is too fine grained for generating a reasonable model. In such a case, filtering and aggregation of events is necessary to prepare the log for a mining algorithm. On the other hand, a mined model usually has to be validated and possibly complemented by the teacher.

A crucial algorithmic problem known from workflow mining is that, in practical settings, we have to assume that not all possible learning sequences have been recorded in a log file, i.e. that logs are incomplete. For such log files the presented mining approach has to be extended. Heuristics can help to infer the

missing sequences and to integrate them into the process model. For the control flow perspective, existing methods from the area of process mining can be used [10]. For the role diagrams we plan to use methods from the theories of structural equivalence and generalized block modeling [39] where missing information is penalized and a solution with minimal penalties can be used for the generalized role model. Moreover, it is possible that a log contains noise, i.e. behavior which should not be regarded in a process model. We think that the problem of noise can be handled by the discussed categorizations of event logs.

Finally, concerning the presented mining approach, the application of the simplification rule for reducing the mined role models has not yet been discussed in detail. In particular, the assisting technique of changing state transitions has to be done manually so far. Automatizing of this procedure is our most immediate topic of further research.

References

1. Barron, B.: When Smart Groups Fail. Journal of the Learning Sciences 12(3), 307–359 (2003)
2. Schank, R., Abelson, R.: Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures. L. Erlbaum (1977)
3. O'Donnell, A., Dansereau, D.: Scripted cooperation in student dyads: A method for analyzing and enhancing academic learning and performance. In: Interaction in Cooperative Groups: The Theoretical Anatomy of Group Learning. Cambridge University Press (1992)
4. Weinberger, A., Ertl, B., Fischer, F., Mandl, H.: Epistemic and social scripts in computer-supported collaborative learning. Instructional Science 33(1), 1–30 (2005)
5. Rawlings, A., van Rosmalen, P., Koper, R., Rodriguez Artacho, M., Lefrere, P.: Survey of Educational Modelling Languages. In: EMLs 2002 CEN/ISSS Learning Technology Workshop (2002)
6. Koper, R., Tattersall, C. (eds.): Learning Design - Modelling and implementing network-based education & training. Springer, Berlin (2005)
7. van der Aalst, W.M.P., van Hee, K.: Workflow Management: Models, Methods, and Systems. MIT Press (2002)
8. Oberweis, A.: Person-to-Application Processes: Workflow Management. In: Process-Aware Information Systems, pp. 21–36. Wiley (2005)
9. Weske, M.: Business Process Management – Concepts, Languages and Architectures. Springer, Heidelberg (2007)
10. van der Aalst, W.M.P.: Finding Structure in Unstructured Processes: The Case for Process Mining. In: ACSD 2007, pp. 3–12. IEEE (2007)
11. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Process Mining Based on Regions of Languages. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 375–383. Springer, Heidelberg (2007)
12. Song, M., van der Aalst, W.M.P.: Towards comprehensive support for organizational mining. Decision Support Systems 46(1), 300–317 (2008)
13. IMS Global Consortium: IMS Learning Design XML Binding 1.0 Specification (2003)

14. Martel, C., Vignollet, L., Ferraris, C., David, J., Lejeune, A.: Modelling collaborative learning activities in e-learning platforms. In: ICALT 2006, pp. 707–709. IEEE (2006)
15. Rodriguez Artacho, M.: PALO Language Overview. Technical report, Universidad Nacional de Educacion a Distancia (2002)
16. Harrer, A., Hoppe, U.: Visual Modelling of Collaborative Learning Processes – Uses, Desired Properties, and Approaches. In: Handbook of Visual Languages for Instructional Design: Theory and Practices. Information Science Reference, pp. 281–298 (2008)
17. Harrer, A., Malzahn, N., Wichmann, A.: The Remote Control Approach - An Architecture for Adaptive Scripting across Collaborative Learning Environments. Journal of Universal Computer Science 14(1), 148–173 (2008)
18. Vogten, H., Martens, H., Nadolski, R., Tattersall, C., van Rosmalen, P., Koper, R.: CopperCore Service Integration - Integrating IMS Learning Design and IMS Question and Test Interoperability. In: ICALT 2006, pp. 378–382. IEEE (2006)
19. Bote Lorenzo, M., Hernández Leo, D., Dimitriadis, Y., Asensio Pérez, J., Gómez Sánchez, E., Vega Gorgojo, G., Vaquero González, L.: Towards Reusability and Tailorability in Collaborative Learning Systems Using IMS-LD and Grid Services. Advanced Technology for Learning 1(3), 129–138 (2004)
20. Harrer, A., Malzahn, N.: Bridging the Gap - Towards a Graphical Modelling Language for Learning Designs and Collaboration Scripts of Various Granularities. In: ICALT 2006, pp. 296–300. IEEE (2006)
21. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow Resource Patterns: Identification, Representation and Tool Support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)
22. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-Based Access Control Models. Computer 29(2), 38–47 (1996)
23. Tan, K., Crampton, J., Gunter, C.: The Consistency of Task-Based Authorization Constraints in Workflow Systems. In: CSFW, pp. 155–169. IEEE (2004)
24. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers 8(1), 21–66 (1998)
25. Desel, J.: Process Modeling Using Petri Nets. In: Process-Aware Information Systems, pp. 147–177. Wiley (2005)
26. Jensen, K.: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Monographs in Theoretical Computer Science, vol. 1-3. Springer, Heidelberg (1992, 1994, 1997)
27. van der Aalst, W.M.P., Moldt, D., Wienberg, F., Valk, R.: Enacting Interorganizational Workflows using Nets in Nets. In: Workflow Management Conference, pp. 117–136 (1999)
28. Valk, R.: Petri Nets as Token Objects: An Introduction to Elementary Object Nets. In: Desel, J., Silva, M. (eds.) ICATPN 1998. LNCS, vol. 1420, pp. 1–25. Springer, Heidelberg (1998)
29. Juhás, G., Kazlov, I., Juhásová, A.: Instance Deadlock: A Mystery behind Frozen Programs. In: Lilius, J., Penczek, W. (eds.) PETRI NETS 2010. LNCS, vol. 6128, pp. 1–17. Springer, Heidelberg (2010)
30. van der Aalst, W.M.P., van Dongen, B.F., Günther, C.W., Mans, R.S., de Medeiros, A.K.A., Rozinat, A., Rubin, V., Song, M., Verbeek, H.M.W(E.), Weijters, A.J.M.M.T.: ProM 4.0: Comprehensive Support for *Real* Process Analysis. In: Kleijn, J., Yakovlev, A. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 484–494. Springer, Heidelberg (2007)

31. IDS Scheer: ARIS Process Performance Manager, http://www.ids-scheer.com
32. Nathan, M., Koedinger, K.R., Alibali, M.: Expert blind spot: When content knowledge eclipses pedagogical content knowledge, AERA, Seattle (2001)
33. Bergenthum, R., Desel, J., Harrer, A., Mauser, S.: Learnflow mining. In: DeLFI. LNI, vol. 132, pp. 269–280. GI (2008)
34. Reimann, P., Frerejean, J., Thompson, K.: Using Process Mining to Identify Models of Group Decision Making in Chat Data. In: Proceedings of Computer-supported Collaborative Learning, pp. 98–107 (2009)
35. Pechenizkiy, M., Trcka, N., Vasilyeva, E., van der Aalst, W.M.P., De Bra, P.: Process Mining Online Assessment Data. In: Proceedings of Educational Data Mining, pp. 279–288 (2009)
36. Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R.: The Cognitive Authoring Tools (CTAT): Preliminary Evaluation of Efficiency Gains. In: Ikeda, M., Ashley, K.D., Chan, T.-W. (eds.) ITS 2006. LNCS, vol. 4053, pp. 61–70. Springer, Heidelberg (2006)
37. Hopcroft, J., Ullman, J. (eds.): Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
38. Process Mining Group Eindhoven Technical University: ProM Homepage, http://is.tm.tue.nl/~cgunther/dev/prom/
39. Doreian, P., Batagelj, V., Ferligoj, A.: Generalized Blockmodeling. Structural Analysis in the Social Sciences, vol. 25. Cambridge University Press (2005)