The MILL – Method for Informal Learning Logistics

Andreas Faatz, Manuel Goertz, Eicke Godehardt, and Robert Lokaiczyk

SAP Research
CEC Darmstadt
64283 Darmstadt, Germany
firstname.lastname@sap.com

Abstract. The paper presents the MILL – a system, which supports planning of training measures. Its background technique, task-competency modeling, is based on formal concept analysis as an indirect and qualitative way of determining the abilities of learners. In that context, the fulfillment or failure of a work task is the indicator of a set of necessary competencies. The core of the presented approach is a matrix structure – a formal context – which has tasks as labels for its rows and competencies labeling its columns. This matrix is the basis for defining formal concepts, which can be ordered in a lattice for navigation and systematic decision support on training measures in an organisation.

1 Introduction

Informal learning means learning activities without a pre-defined support by curricula, textbooks or other classical didactical material imposing a learning path. This paper describes a method for informal learning logistics. It aims at managing training measures in informal learning environments. This particularly means learning environments that are closely embedded into the working place and which are related to the logical and temporal order of tasks (i.e. the workflow) a worker needs and wants to fulfill on the job. Typically, the nature of this work is knowledge-intensive.

The informality of a training measure becomes clear by the fact, that the effect of some or all of the training measures can often only be monitored by the outcome of working tasks. This contrasts with the situation at school or university, where the taught competencies to can be tested by exams. The system described in this paper enables the detection of positive or negative expected consequences of particular training measures in workplace-embedded learning. Additionally, it shows how to plan individual training measures for the worker corresponding to the tasks s/he is responsible for.

The approach to these questions is a mapping of well-ordered and well-structured tasks to a task-competency model. This conceptual structure is queried systematically by the system to exploit the temporal order and conditional interdependency of tasks. The querying creates a feedback on potential positive or negative consequences for the task structure and the progress of the workflow.

The achieved results allow a human resources manager to precisely determine competencies, which should be achieved by training measures. We abbreviate the system to be presented with the MILL – Method for Informal Learning Logistics.

The paper is organised as follows. We start with related work and identify the basic requirements on our system. In section 3 we explain the necessary elements of formal concept analysis and Petri nets, which will be the building blocks of the MILL. Section 4 is dedicated to a detailed description of the system, especially regarding the interplay between the task-competency model reflected as a formal concept lattice and the workflow formalised as a Petri net. Section 5 will present our conclusions and a brief outlook on future work.

2 Related Work and basic requirements

Task-competency modeling [1] is based on formal concept analysis. It is an indirect and qualitative way of determining the abilities of learners. The fulfillment of a task is the indicator of a set of necessary competencies. The core is a matrix structure, the so-called formal context [2], which has tasks as labels for its rows and competencies labeling its columns. This matrix is the basis for defining so-called formal concepts [2], which can be ordered in a lattice for navigation.

Ley et al. show two application scenarios in the domain of informal learning [3]. First, a task included in a workflow, for example the preparation of a document, is an indicator of the learner's competencies. Example competencies might be language capabilities, ability to abstract, ability to structure a topic and particular domain knowledge. Thus, by judging about the quality of a task output (in this example: the quality of a document) we gain a detailed picture on the competencies which enable a person to fulfill the task. Ley argues, that the resulting conceptual structures (intensions of the concepts are tasks, extension of the concepts are competencies) allow planning the training measures of an organization. In a second scenario the authors establish the lattice from the competencies and taks and present it in a graphical representation for navigation. This enables self-directed learning, where the user can easily determine, which documents match best for his/her training needs.

Although the authors present a flexible formal basis for conceptual structures derived from competencies and taks, the technical progress with respect to the core ideas of formal concept analysis does not become clear – except by the advances made by the approach of the former knowledge space theory due to Korossy [1], which identifies sets of competencies in a non-quantitative way. The authors do not explain systematic browsing or algorithms for browsing through the conceptual structures or typical queries, which might support the planning of training measures. The task-competency model is not linked to other parts of a learning environment or workplace. The interpretation of the concept lattice and its mapping on a temporal sequence of tasks or interdependent tasks is an open question we solve by the techniques of the MILL.

From the point of view, which enhances Petri nets (as an example for formally modeled workflows) by conceptual structures there is prior work in describing the elements of Petri nets semantically [4]. Koschmieder et al. express the whole Petri net as constructs in the web ontology language OWL [5]. Koschmieder's aim is a modularization and recombination of workflows expressed by Petri nets across several business units or organizations. The semantic description is not applied to planning of and reasoning about tasks or developing competencies by training measures.

With the gaps of the related work in mind we state the following main requirements for a system for planning training measures:

- The system should consider a learning situation outside the classroom and thus also apply to training in organisations or during the work process. The organisational workflow must be part of the system's underlying model.
- The system should be able to assist planning based on the interdependencies of tasks as well as on the interdependencies of the competencies, which are characteristic for a task.
- The assistance provided by the system should be formalised in a way, which helps the planner to understand the decisions or suggestions made by the system. Especially the consequences of training measures (or the consequences of left-out training measures) should be trackable for the user. The user might be a planner as well as a person, who gets trained.

3 Existing background techniques

The following main sections of the paper will give a survey of the relevant background techniques, namely formal concept analysis capturing the task-competency model and Petri nets as a universal formalism for descriptions of organisational workflows. Moreover, we continue explaining the innovative core system (MILL) and end with an example. Throughout our explanations, we work with the definition of a task as 'an action performed to reach a particular goal' [6].

Notation: the paper uses the following notations for mathematical constructs:

- sets are italic capitals or denoted in the usual way with comma-separated elements between braces: $\}$ and $\{$.
- $\mathbf{I}$ is a relation between tasks and competencies (in formal concept analysis between general objects and properties)
- small italic Latin characters denote tasks and competencies (in general formal concept analysis between objects and properties)
- small Greek characters denote Petri nets
- $'$ is the derivation operator for formal contexts
- formal concepts are written as pairs of sets or as bold italic capitals
- if $*$ is attached to a set notation, it denotes a subset of the set originally notated
- a set followed by $\sim time$ is partially ordered regarding some time scale, similarly $\sim cost$ stands for partial order regarding some cost scale
- $\boldsymbol{l}$ is a list of insufficient competencies

Formal Concept Analysis (FCA) is a theory of conceptual structures (lattices), which result from the simultaneous reasoning about objects and their properties. FCA was founded by Wille [2]. The derivation of the concept lattice is based on tables called formal contexts, where an entry (a cross) indicates, if a property is fulfilled or not. We define along with Wille's theory the notion of a Formal Context.

Definition 1. *Formal Context: let G be a set of objects and M a set of properties. Let $\mathbf{I}$ be a binary relation indicating, which object from G fulfills which property from M. We write $g\mathbf{I}m$, if an object g from G fulfills property m from M. G, M and $\mathbf{I}$ together are called formal context $(G, \mathbf{I}, M)$.*

Table 1 shows an example of a formal context named task-competency. Objects a_1 through a_6 denote tasks, where particular documents of a software engineering process must be written or completed. Thus we will simply refer to these tasks as 'documents 1 through 6' in the following explanations. The attributes (denoting the columns) are abbreviations for language (lan), abstraction (abs) and Unified Modeling Language (UML). A checked box means, that for writing the document in the specific row, the marked competencies (e.g. language, abstraction or knowing UML) have to be fulfilled. For instance, for document 1 all these competencies are necessary, but for document 3 only language and UML are required. (The figure was generated with the online tool JaLaBA, see http://maarten.janssenweb.net/jalaba/JaLaBA.pl). Throughout the document we will use Arabic numbers for indexing to temporally order tasks (the task with index number 1 denotes the first one in the workflow). Concurrent tasks would be denoted for example as 1 (concurrency to task 1).

Table 1. Formal context

	Competencies		
Task	lang	abs	UML
a_1	$\surd$	$\surd$	$\surd$
a_2		$\surd$	$\surd$
a_3	$\surd$		$\surd$
a_4	$\surd$	$\surd$	$\surd$
a_5	$\surd$		$\surd$
a_6	$\surd$	$\surd$	

We define a derivative operator $'$ for subsets $X \subseteq G$ and $Y \subseteq M$ mapping the objects (properties respectively) from X (Y, respectively) to those properties from Y (objects from X, respectively), which fulfill the relation $\mathbf{I}$ for at least all objects (properties respectively) from X (Y, respectively). For the derived sets we write X' and Y', respectively. The construction of a concept lattice is possible by applying several derivations to sets of objects or properties. Note: X

is always a subset of X'' and X' equals X'''. We can define the notion of a *formal concept* of a formal context $(G, \mathbf{I}, M)$ is a pair (A, B) where A is a subset of G and B is a subset of M and $A = B'$. We say that two formal concepts (C, D) and (E, F) fulfill the sub-concept relation $\leq$, if and only if C is a subset of E. Here $(C, D) \leq (E, F)$ reads "(C, D) is a sub-concept of (E, F)". $\geq$ would denote the inverse relation of $\leq$, and $(C, D) \geq (E, F)$ reads "(C, D) is a super-concept of (E, F)". Table 2 shows the concepts resulting from the above task-competency context (computed with JaLaBA). Elements of the first set in the pair forming a formal concept are called extension of the concept, elements of the second set are called intension of the concept.

Table 2. Formal concepts

	Formal Concepts of task competency
$\boldsymbol{A}$	$< a_1, a_4, \{$language, abstraction, UML$\} >$
$\boldsymbol{B}$	$< a_1, a_2, a_4, \{$abstraction, UML$\} >$
$\boldsymbol{C}$	$< a_1, a_3, a_4, a_5, \{$language, UML$\} >$
$\boldsymbol{D}$	$< a_1, a_4, a_6, \{$language, abstraction$\} >$
$\boldsymbol{E}$	$< a_1, a_2, a_3, a_4, a_5, \{UML\} >$
$\boldsymbol{F}$	$< a_1, a_2, a_4, a_6, \{$abstraction$\} >$
$\boldsymbol{G}$	$< a_1, a_3, a_4, a_5, \{$language$\} >$
$\boldsymbol{H}$	$< a_1, a_3, a_4, a_5, a_6, \{\} >$

A visualization generated with ToscanaJ (see ToscanaJ project page at sourceforge.net) of the resulting lattice is shown inFigure 1.

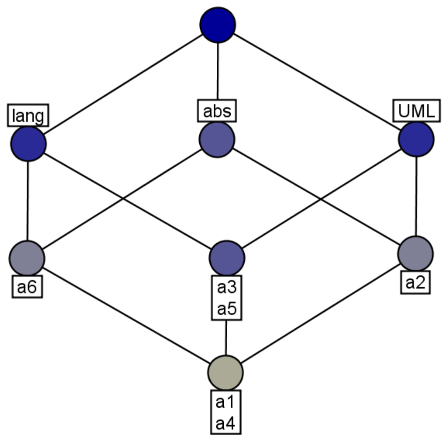


Fig. 1. Concept lattice

Concepts are shown as colored circles, super-sub-conceptual relationships are shown as lines. A line between a darker and a brighter concept means, that the brighter one is a sub-concept of the darker one. The concept at the very top refers to concept $\boldsymbol{H}$, the concept with the label 'lang' refers to concept $\boldsymbol{G}$, the one with the label 'abs' refers concept $\boldsymbol{F}$, the concept with the 'UML' label refers concept $\boldsymbol{E}$, the concept with the label 'a6' below it refers concept $\boldsymbol{D}$, the concept with the label 'a3 a5' below it refers concept $\boldsymbol{C}$, the concept with the label 'a2' below it refers concept $\boldsymbol{B}$, the concept with the label 'a1 a4' below it refers concept $\boldsymbol{A}$. The intension of a concept can be read by following all paths, where any step in the path end in a darker concept and collecting all labels in top of a circle (i.e. concept), the extensions can be read by following paths the other way around and collecting all labels below a circle (i.e. concept).

A Petri net consists of tokens, transitions and places as well as arcs, which connect places and transitions. The distribution of tokens indicates the state of the Petri net. If a transition is enabled, all places pointing to the transition must be filled with tokens. Tokens can be of different color. This refers to typed events. For example, in a system different colors might refer to different users of a system and the respective state of their workflows. Different colors of tokens might also result from several cases or applications using the same workflow (i.e. here: Petri net). An important characteristic of Petri nets is their openness regarding the way transitions are fired. Tokens at incoming places only enable the transition. There is the possibility of assigning time information to each transition. Time information can be attached to each transition as a real number characterizing the behavior of the transition. The number means e.g. the amount of milliseconds the transaction will need after its enabling to produce its outgoing tokens. A detailed introduction to Petri nets can be found in [7].

Van der Aalst [8] also showed why Petri nets are a good candidate for organisational workflow modeling. First, their graphical expression has a clear formal meaning. Moreover, a Petri net is always a state-based notation in contrast to an event-based notation. The enabling of transitions can be seen at a glance and is not due to any kind of (semi-formal or informal) interpretation. The state-based notation also eases the expression of concurrency. A process leaving the system completely can simply be expressed by removing all its (colored) tokens. Finally, a multitude of mathematical formalisms exist that allow the checking of the properties of a Petri net. The formalisms can directly be applied to Petri nets expressing workflows and for proving the soundness of a workflow. For example it can be shown, that all tasks are free from the danger of falling into a deadlock situation.

Van der Aalst has shown that Petri nets are capable of formalizing current process-aware information systems. Figure 2 shows an example Petri net without tokens. The circles indicate places (for instance 'indirect users found'), the quadratic shapes transitions (for example 'prepare developers list'). Tokens would be indicated by colored circles centered at places. The arcs starting at a place and ending at a transition can be understood as 'is necessary condition for', arcs starting at a transition and ending at a place can be understood as 'triggers state'. Note that this is rather a snapshot of a larger Petri net, as in real-world applications the places at the very left would be connected to a starting transition.

4 Description of the core system

Prerequisites: the MILL-system has the following prerequisites and components:

- a workflow is modeled as or transformable to a Petri net,
- for the sake of our explanation we assume a single worker in a single workflow,
- a task-competency-matrix exists, thus the conceptual structures like in Figure 1, Table 1 and Figure 2 exist,
- the tasks are the same or a super-set of the tasks corresponding to the transitions in the Petri net.

For example, relating these prerequisites to the examples of Figure 1 and Figure 2 would mean, that besides the modeling expressed by the two figures concepts $\boldsymbol{A}$ through $\boldsymbol{H}$ would correspond to transitions (= tasks) from Figure 2. Thus Figure 2 without the transition 'pool facilitating stakeholders' and without the arcs incoming in and outgoing from it would fulfill the prerequisites, if additionally the rectangular shapes were foreseen with the elements $\{\boldsymbol{A}, \ldots \boldsymbol{H}\}$.

If the tasks of the workflow are a proper subset, the task-competency matrix and the resulting formal context can be restricted to those tasks, which are actually contained in the workflow.

The operations of the MILL-system starts at the moment, when a task t of a worker fails (case A) or is judged to be fulfilled in a non-sufficient way (case B). This situation might occur for reasons, which originate outside of the organization (e.g. customers refuse the outcome of a work package) or for internal reasons (e.g. internal reviews). The workflow, i.e. the Petri net with all its tokens at the current places is frozen before (case A) or directly after (case B) the transition is fired. Two ratings supporting the identification of competencies to be improved start immediately and (potentially) in parallel: a competency-based rating and a task-based rating triggering a planning procedure. The competency-based rating is a direct one, the task-based rating a more indirect one. Both ratings return a set of tasks D, which we call critical tasks.

Competency-based rating: the competencies, which are necessary to fulfill the task t, are checked against the competencies of the worker. The MILL-system either has access to such competency ratings for each worker or prompts an evaluator to judge about the competencies. Up to this point, competency-based rating resembles and formalizes the ideas of [3]. From this point on, all further steps and techniques we introduce (for competency-based rating, task-based rating and beyond) are innovative.

The insufficient competencies are returned as an ordered list $\boldsymbol{l}$ (if possible, ordered from the worker's worst competency to best but still insufficient competency). An example for such a list could be: $<$ language, UML $>$. The critical

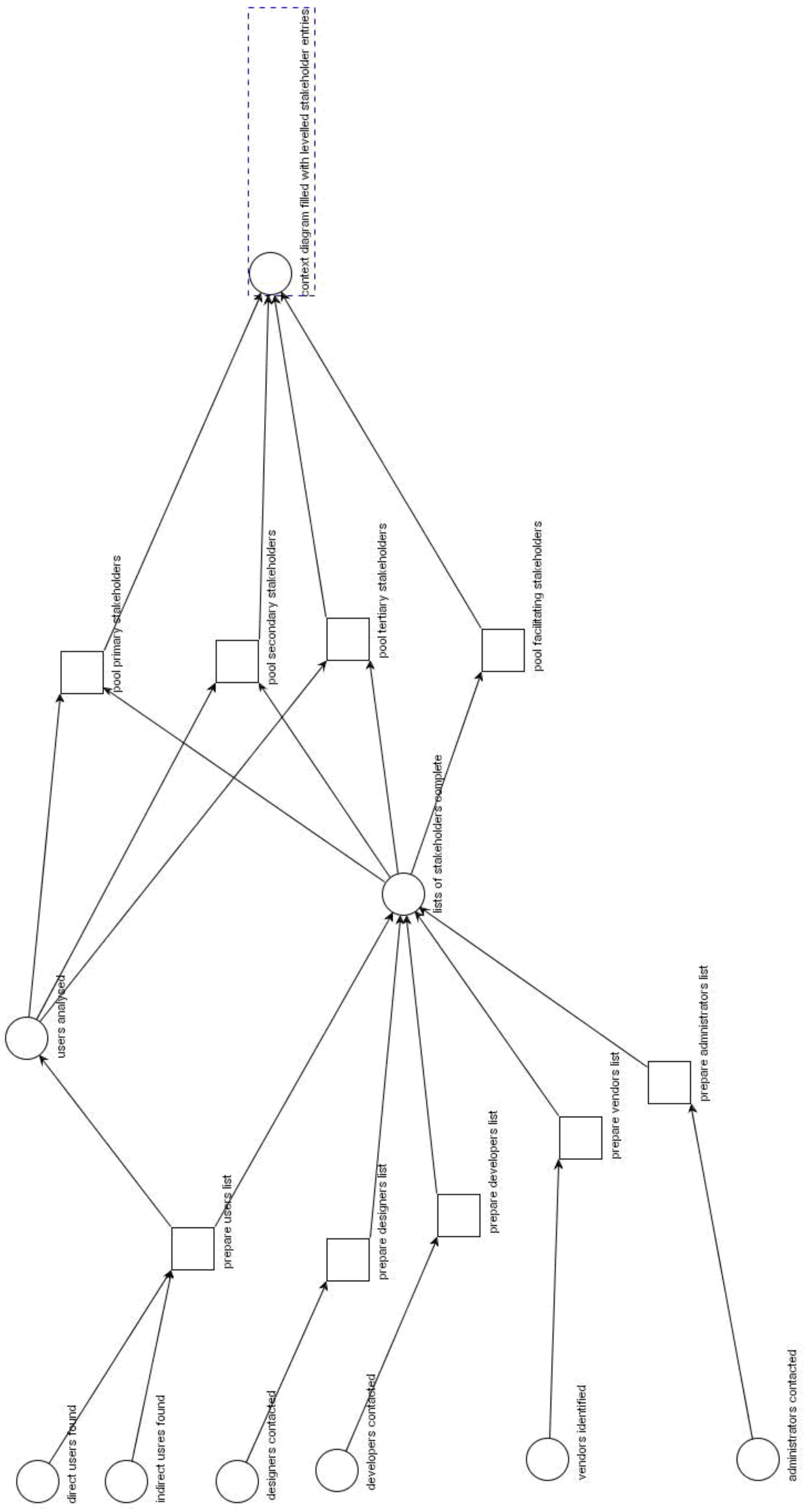

context diagram filled with levelled stakeholder entries
pool primary stakeholders
pool secondary stakeholders
pool tertiary stakeholders
pool facilitating stakeholders
users analysed
lists of stakeholders complete
prepare users list
prepare designers list
prepare developers list
prepare vendors list
prepare administrators list
direct users found
indirect usres found
designers contacted
developers contacted
vendors identified
administrators contacted

Fig. 2. Petri Net model

tasks D are obtained as the result of browsing down the concept lattice, starting from the most upper (more grey) concepts, which have an insufficient competency as intension. By browsing we mean moving down along the lines, which indicate a relation $\geq$ or $\leq$ between two concepts. An example of these lines can be found in Figure 1 between the concept with the label 'a6' and the concept with the label 'abs'. The browsing through the concepts starts from the top of the concept lattice and from the worst to the best – but still insufficient – competency. This action collects all tasks from the respective extensions except the ones below concepts, where a competency (to be identified as the label heading the circle belonging to the concept), which is either sufficient or not explicitly on the list of insufficient competencies, can be retrieved.

Task-based rating: In this case, $\boldsymbol{l}$ does not necessarily exist. The task t is evaluated without evaluating the single competencies contributing to it. The MILL-system might have access to ratings of tasks in the past. Critical tasks are future tasks in the workflow, which cannot be performed because of lacking competencies. The task-based rating generates other critical tasks:

- promptly as the extension of the concept, which is generated by applying the derivation operator to t twice (i.e. t''). This means the critical tasks D are those tasks in the future from the extension of the most special (brightest in the grey-scaled layout of Figure 1) concept containing t. Consider from the example in Figure 1 and Table 2, that for instance $t = a_1$. Then the critical tasks D would be a_1 and a_4.
- historically by comparing the current failure of t to failed tasks in the past. If there was a failed task s in the past and no training measure related to the competencies in its intension, then extension of the most general concept $\boldsymbol{Q}$ with $\boldsymbol{Q} \leq (t'', t')$ AND $\boldsymbol{Q} \leq (s'', s')$. For instance, let $t = a_4$ and $s = a_2$ in the example in Figure 1 and Table 2. Here, s is assumed to be in the past as $s < t$. Furthermore, $t'' = a_1, a_4$ and $s'' = a_1, a_2, a_4$. Then $D = a_1, a_4$.
- If there are more than two failed tasks from the past, the critical (future) tasks D can be determined by repetition of this procedures.

Ordering and selecting critical tasks: the necessary training measures to improve the overall performance of an organization are now prioritized by temporal and conditional aspects, which can be derived from the workflow expressed by the Petri net.

If the set of critical tasks D results from competency-based ranking, then it can be ordered by assigning costs (that means: a positive real number) to each lacking competency from the list $\boldsymbol{l}$. The costs should increase with higher insufficiency of the lacking competency. Summing over the costs of all competencies necessary for performing a particular critical task in D yields total costs of missing competencies. The higher these costs, the more likely the particular critical task will not be performed by the worker. This fact provides a basis for deciding about training measures. For instance, if the costs reflect the training costs, the planner might decide to chose the least expensive training measures first. Another potential cost function is the cost of scheduling another person to fulfill the task.

If D is the outcome of a purely task-based ranking, then there is no similar ordering regarding costs of missing competencies, as this innovative way of obtaining D judges taks based on competencies. Thus there might be other competencies, which are not measurable in a single step task-based rating only. Task-based rating rather provides immediate information on other surely critical tasks.

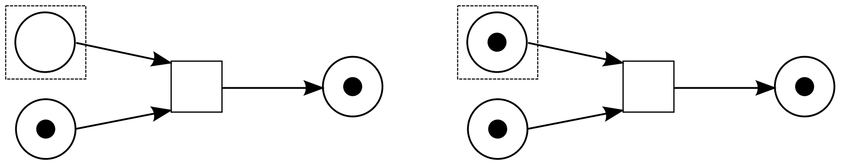
In both cases an alternative ordering of D by temporal aspects is possible. The tasks (corresponding formally to transitions) in the Petri net are partially ordered in the sense that for some pairs of tasks (transitions) it is possible to state, which task (transition) will be enabled executed after the other one. In the example based on Figure 2, some of the tasks (transitions) can be ordered temporally as follows: 'prepare users list' comes before 'pool primary stakeholders', 'prepare administrators list' comes before 'pool facilitating stakeholders'. In cases of concurrency, Petri nets with time information are even able to resolve the concurrency and order the tasks along a timeline. From this ordering point of view, a critical future task, which approaches earlier in the workflow, could get priority in comparison to a later critical task from the planning point of view.

In the example from Figure 2, all tasks described as 'pool ' are concurrent and all tasks described as 'prepare ' are concurrent. If for example 'prepare users list' and 'prepare designers list' were attached with time information, that the firing of the transition needs 3 days for 'prepare users list' and 1 day for 'prepare designers list', then the order created would place 'prepare designers list' before 'prepare users list'.

No matter which alternative is chosen, the ordering step of the MILL-system results in a partially ordered set $D \sim time$ (if it is partially ordered by temporal aspects) or $D \sim cost$ (if it is partially ordered by costs). This set already contains not only the critical tasks, but also a priority, either driven by the strength of the criticality or by time issues. Before we proceed with the final selection of critical tasks, we remark that D might also result from a mixture between or union of task-based and competency-based rating. In this case, the partial ordering will be created as $D \sim time$.

The last step in determining prioritized training measures is applying van der Aalst's second soundness criterion [8] to simple transformations of the workflow captured by the Petri net. This soundness criterion investigates, if a procedure expressed by a Petri net terminates eventually. This might be generalized to a criterion, that the procedure terminates in time due to a real-time condition. An example of a Petri net, which does not terminate, is depicted in Figure 3(a). This Petri net always misses a token to proceed, whereas the Petri net in Figure 3(b) will terminate. Nevertheless, even the Petri net from Figure 3(b) would hurt the real-time criterion, if for instance the firing of the transition takes one day instead of one hour, which could be an example time threshold.

Let β be a Petri net capturing the workflow. Then let $\beta(D^*)$ denote a Petri net equal to β except for the fact, that all transitions, which correspond to a



(a) Petri Net with unfired state (b) Petri Net with fired state

Fig. 3. Petri nets

subset D^* of D, are absolutely disabled. An example of such a $\beta(D^*)$ is the Petri net resulting from adding incoming absolutely empty places to all transitions corresponding to tasks from D. We call this example of a Petri net with disabled transitions partially disabled. If for example β is the Petri net from Figure 2 and D^* turns out to be the single-element set 'pool primary stakeholders', then $\beta(D^*)$ would be Figure 2 with an additional token-less place, which has an arc to the transition 'pool primary stakeholders' as its only connection to the example Petri net β.

If $\beta(D^*)$ obeys the soundness or real-time criterion for all subsets D^* of D, then the MILL suggests to train the tasks from $\boldsymbol{l}$, which cause the endangerment of the earliest (or most costly) critical task from $D \sim\!time$ (or $D \sim\!cost$). If $\boldsymbol{l}$ does not exist, but still all $\beta(D^*)$ obey the soundness or real-time criterion, the MILL suggests training of competencies from the intension of the most general concept $\boldsymbol{Q}$ which is a sub-concept to all $|D|$ concepts generated by twofold derivation of a single-element subset of D. If this intension is empty, the derivations and the computing of $\boldsymbol{Q}$ are repeated on the basis of successively removing late tasks (due to $D \sim\!time$) or expensive tasks (due to $D \sim\!cost$) from D, until $\boldsymbol{Q}$ is nonempty. The MILL also informs the planner, that the suggested training measure is not necessary for keeping the workflow sound.

In all other cases with non-sound $\beta(D^*)$ for some D^*, the MILL passes a warning to the planners, saying that the workflow as a whole is critical. If there are one or several subsets $D^*(1), \ldots, D^*(n)$ of D, for which the resulting partially disabled Petri nets do not obey the soundness or real-time criterion, then the MILL operates through the following steps:

– determining minimal subsets of each $D^*(1), \ldots, D^*(n)$ in the sense, that for each proper subset of each $D^*(1), \ldots, D^*(n)$ the corresponding partially disabled Petri net would still obey the soundness criterion (real-time or van der Aalst's).
– determining the intension(s) of the most abstract concept(s), which contain a single task from the respective $D^*(1), \ldots, D^*(n)$ in their extension. Determine the latest of these single tasks for $D^*(1), \ldots, D^*(n)$ respectively, if there is more than one formal concept with the same most-like level of abstraction.
– for each set $D^*(1), \ldots, D^*(n)$ the MILL proposes all the intensions of the aforementioned most abstract concepts.

If in addition $\boldsymbol{l}$ exists, the MILL only suggests training measures corresponding to those insufficient competencies. The strategy described for cases with non-sound $\beta(D^*)$ for some D^* corresponds to the idea of reserving enough time in the organization to train competencies, which will in their end-effect keep the workflow possible and alive. Following this strategy, the resulting intensions are again ordered by the MILL proceeding through the following steps:

– for each of the retrieved intensions identify the earliest task (regarding $D \sim time$) in their respective extension and denote its time.
– the intension with the time closest to the current failed task is proposed by the MILL as the training measure.

The proposition is based on this order, because it allows for step-by-step safe traversing of competencies (contained in the intensions) to be sure to cover tasks (contained in the extensions), which would stop the future workflow in case of failure. To keep this safety, the MILL is applied after enabling and before firing a transition, as long as there are current or past tasks, which failed and still could fail due to the task-competency context and an update of the MILL-system. All time-dependent decisions are then relative to the current task (i.e. transition). We conclude with the description of a potential update mechanisms.

Update: For the update phase, the frozen state of the workflow is removed. That means, that the workflow is continued: the next enabled transition(s) is (are) fired. The update of the MILL-system can be - up to decision of the organization, where it is implemented - foreseen with one or several paradigms:

– the MILL might work pessimistically with a virtual list $\boldsymbol{l}$ containing all competencies except the ones in the derivation of successful tasks from the past.
– alternatively and potentially complementarily, whenever a competency is taught, the MILL-system triggers a new task-competency matrix, where the relation $\mathbf{I}$ between the competency just taught is removed from the task-competency matrix or removed from the (virtual or real) list $\boldsymbol{l}$.
– also alternatively and potentially complementarily, an update could also include outdating-functions for competencies - for instance, if no task with an application of language competencies has to be performed after the language competencies were trained. The outdate is implemented as insufficient competency re-appearing on $\boldsymbol{l}$ or as re-appearing relation $\mathbf{I}$ in the task-competency matrix.

Finally, note that the MILL supports planning of training measures. It gives no direct clue to alternative paths through the workflow, planning of working tasks or workflow optimization.

5 Conclusion

We presented the MILL as a novel approach to a system, which connects the task-competency approach to well-structured workflows, which are typical for organizations (enterprises, administrations et cetera). The competencies are all abilities or skills of the workers, which support tasks in the workflow and which in a narrower sense can be taught (e.g. programming skills, language skills etc.) or in a broader sense be developed (e.g. management skills, communication skills etc.). Our approach matches the the three central requirements from Section 2 in the following ways:

- Classical knowledge space theory and applied competency-performance structures [1] operate in a classroom situation, thus its applications focus on adaptive e-Learning systems [9], which try to cover a landscape of competencies while the learner interacts with these systems. The interdependencies of tasks (in many cases the tasks are test items) is not in the focus of this (virtual and non-virtual) classroom applications (this means: the order of test items is irrelevant). The MILL is focused on real-world tasks in an organization and its aim is to cover the competencies necessary for concrete future actions of the learner.
- The temporal interdependency of tasks, which are past, current or future activities of a worker is systematically exploited by the system. Prior work has not established or exploited any orders and relations of tasks. The MILL distinguishes past and future tasks as well as costs of lacking competencies.
- The consequences of missing or improving competencies for future and repetitive tasks are formalized. This formalization is a novel mapping of task-competency modeling to workflow analysis. Prior work has not coupled training activities with consequences for the actual workflow. The MILL also performs a second check after critical future tasks are identified.
- The MILL is centered around a systematic browsing of a concept lattice resulting from the task-competency structure. Prior work [3] focuses on the task-competency structure and resulting learning paths itself; its traversing by (temporally and conditionally) structured tasks from a workflow was not formalized, yet. The MILL innovates the view on the tasks and competencies as a dyadic one: it is possible to reason from failed tasks and from lacking competencies.
- The approach includes rich diagrammatic structures, which can be used as an explanation of the system's decisions. In particular, the MILL gives reason to the priorities of training measures.

Future work will focus on the question, if an extension of the approach considering a fuzzyfication of the task-competency structure might be fruitful. The idea behind such a fuzzyfication is capturing increasing or decreasing competencies over time. This will be challenging work, as there are several approaches to fuzzy formal concept analysis. Another useful extension would be a reasoning mechanism, which considers the competencies of a whole team instead of single workers.

References

1. Korossy, K.: Extending the theory of knowledge spaces: A competence-performance approach. Zeitschrift für Psychologie (1997)
2. Wille, R., Ganter, B.: Formal concept analysis. In: Mathematical Foundations, Springer-Verlag (1997)
3. Ley, T., Lindstaedt, S., Albert, D.: Supporting competency development in informal workplace learning. In: Lecture Notes in Artificial Intelligence. Number 3782 (2005)
4. Koschmider, A., Oberweis, A.: Ontology based business process description. In: Proceedings of the CAiSE05 WORKSHOPS. (June 2005)
5. W3C: Overview and features of owl (2004) www.w3.org/TR/owl-features/.
6. van Welie, M., van der Veer, G., Eliens, A.: An ontology for task world models. In: Proceedings of DSV-IS98, Abingdon UK, Springer-Verlag (1998)
7. Sowa, J.: Knowledge representation: logical, philosophical, and computational foundations. MIT Press (2000)
8. van der Aalst, W.M.P.: The application of petri nets to workflow management. Journal of Circuits Systems and Computers **8**(1) (1998)
9. Hockemeyer, C., Held, T., Albert, D.: Rath-a relational adaptive tutoring hypertext www-environment based on knowledge space theory. In: Proceedings of CALISCE. (1998)