Visual Language Plans - Formalization of a Pedagogical Learnflow Modeling Language

Kerstin Irgang[1] and Thomas Irgang[2]

[1] Human-Centered Information Systems, Clausthal University of Technology,
kerstin.pfahler@tu-clausthal.de
[2] Department of Software Engineering and Theory of Programming, Fernuniversität Hagen,
thomas.irgang@fernuni-hagen.de

Abstract. In this paper we present an approach to support selfregulated learnflows in the collaborative environment Metafora. In this environment students construct Visual Language Plans. Those plans model workflows of learning activities, which the students execute to solve complex learning scenarios across different tools.
Visual Language plans were already used in the context of different pedagogical studies but have no formal syntax or semantics, yet. In this paper, we present the syntax of Visual Language Plans and develop a mapping from Visual Language Plans to Petri net defining semantics. With the help of this semantics, the environment can support the students executing their learnflows. If students execute activities given in a Visual Language Plan which are not enabled in the corresponding Petri net, feedback messages occur guiding the students. Students can refine their Visual Language Plan during execution. If a plan changes the corresponding Petri net model also changes. Analyzing the newly generated Petri net model can help to uncover faulty states of the learnflow model.

1 Introduction

On the one hand, most schools and universities use traditional eLearning systems like Moodle or Ilias. These eLearning systems, which work like Groupware systems in industry, support workflows and offer a lot of evaluation tools, but miss most of the Web 2.0 features and collaboration. On the other hand the students use interactive and collaborative Web 2.0 systems like Facebook or Twitter. New pedagogical approaches try to close this gap and benefit from supporting collaboration between students.

The Metafora project [1] developed a *Computer Supported Collaborative Learning* (CSCL) system [2], which takes up the advantages of social networking technologies. The project was co-funded by the European Union under the 7th Framework Program for R&D, with several partners on the technological and pedagogical side. In the Metafora system, students between 12 to 16 years learn math and science in an enjoyable and selfregulated way, working collaboratively in groups of 3 to 6 members on a complex challenge they have to solve. One aspect of the project was to developed a so-called *Visual Language* [3] for planning and executing of learning activities. One of the main goals of this language is to serve the *Learning to learn together* (L2L2) [4] approach (see e.g. [5]). The publications [6,7] describe the L2L2 approach in detail.

In the Metafora system a group of students gets a challenge which they solve collaboratively. Therefore the students build a Visual Language Plan, consisting of different cards and arcs. This plan describes and documents their approach to the challenge. Teachers can create new challenges. A step of creating a challenge is to select the set of available cards [8]. The Metafora system supports the students in modeling and executing their Visual Language Plan. When the students execute their plan they use it to join collaborative instances of the used tools and document the current state of their execution. When the students finished their plan they reflecting about their process with the help of the plan. Figure 1 shows an example of a Visual Language Plan.

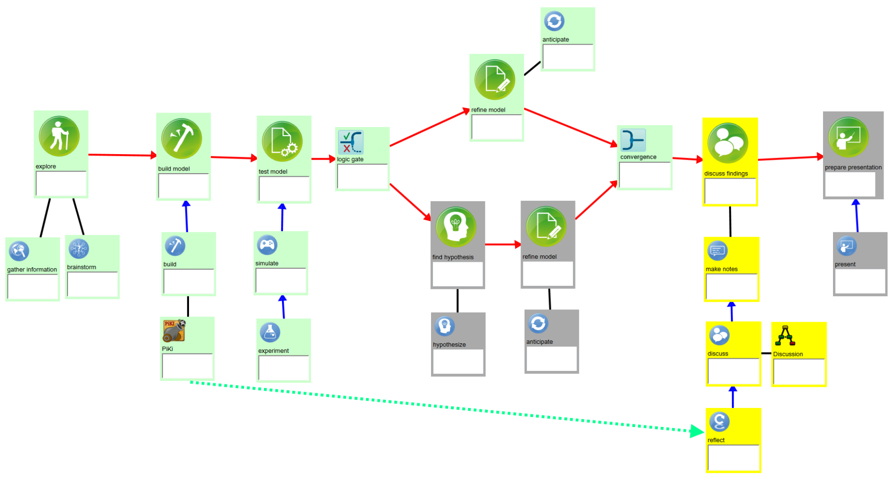

Fig. 1. An example of a Visual Language Plan.

The Visual Language Plans have a graphical representation and consists of nodes and arcs. This nodes are cards. Different types of arcs connecting these cards. A card represents a learning activity. Cards are in one of the three states idle, started or finished. The Visual Language Plan depicted in Figure 1 was developed by students solving the challenge *The bouncing cannon ball* [9]. The students see the bouncing of a cannon ball and simulate how changing variables like angle and speed change the trace of the ball. Students explore the phenomenon with the integrated, game-based domain tool *PiKI* (Pirates of the Kinematics Island). In PiKI students fire cannon balls from a pirate ship to an island trying to hit treasures. To solve the challenge the students create a plan and follow it step by step. In case of the plan depicted in Figure 1 they start with exploring the challenge by gathering for information on bouncing effects and brainstorming ideas. Afterwards the students build a model with their ideas in the domain-tool *PiKI*. In the next step, the students test their model with simulating and experimenting on it. Depending on the results of their test, the students rather go ahead directly to refining their model, or looking for a new hypothesis on the effect of changing variables of the ball if the test failed. Nevertheless, the students discuss their findings afterwards, by reflecting on the previous results in PiKI. The students use the integrated, graphical discussion environment LASAD [10] for their discussion and make notes about the findings. Finally, the students create a presentation and present their results to other groups.

Although the Visual Language is very intuitive, it does not have a formal syntax or semantics. Both are essential requirements to support the students by modeling a Visual Language Plan. The papers [5] and [11] introduce the core elements of the Visual Language, but they give no construction rules. First syntactic rules were already defined in [12]. Now, we fine grain and formalize the syntax of the Visual Language and develop semantics rules. To define the semantics of a Visual Language Plan we give a mapping to a Petri net and exploit the occurrence rule of Petri nets. There are other approaches using Petri nets to model learnflows. The paper [13] considering the teacher as the expert who models a learnflow which students execute. This approach already had the advantage of an executable model, which allowed simulation and usage of a workflow engine, but teachers did not accept it and the students did not understand the model. To overcome this, the Metafora project developed Visual Language Plans as an intuitive graphical representation for learnflows and examined it in classrooms with teachers and students.

We organized the paper as follows. In Section 2, we give the required definitions of Petri nets and in Section 3 we develop a mathematical founded syntax for Visual Language Plans. Section 4 describes the mapping of Visual Language Plans to Petri nets and Section 5 shows our use cases for the new semantics of Visual Language Plans. In Section 6 we sum up our work.

2 Petri nets and their occurence rule

In this paper we use the following notations. With $\mathbb{N}_0$ we denote the non-negative integers, i.e. $\mathbb{N}_0 = 0, 1, 2, \ldots$ and with $|S|$ we denote the cardinality for a finite set S. Petri nets [14] are bipartite graphs of places and transitions which are a good tool to model concurrent systems:

Definition 1 (Petri net). *A Petri net is a 4-tuple $N = (T, P, F, m_0)$, where T is a finite set of transitions, P is a finite set of places, $F \subseteq (T \times P) \cup (P \times T)$ is a finite set of edges and $m_0 : P \rightarrow \mathbb{N}_0$ is a marking . The sets T and P are disjoint, i.e. $T \cap P = \emptyset$.*

In graphical representations, we draw the transitions as squares and the places as circles. If an edge between a place and a transition exists, we draw an arrow. We show the marking for a place with small dots drawn in that place. To define the semantics of a Petri net, we use the preset and postset of a transition.

Definition 2 (Pre- and Postset). *Given a Petri net $N = (T, P, F, m_0)$. The preset of a node $n \in (T \cup P)$ is the set $\bullet n := \{n' \in (T \cup P) \mid (n', n) \in F\}$. The postset of a node $n \in (T \cup P)$ is the set $n\bullet := \{n' \in (T \cup P) \mid (n, n') \in F\}$.*

In a Petri net, enabled transition has only marked places in its preset. Only enabled Transitions can occur. If a transition occurs the marking of the Petri net changes. The transition consumes marks from its preset and produces new marks in its postset.

Definition 3 (Occurrence Rule for Petri nets). *Given a Petri net $N = (T, P, F, m)$. A transition $t \in T$ is enabled, iff for all places $p \in \bullet t : m(p) > 0$ holds. The occurrence of t yields the new marking $m' : P \to \mathbb{N}_0$. This marking is:*

$$m'(p) := \begin{cases} m(p) - 1 \ , & \textit{if } p \in \bullet t \textit{ and } p \notin t\bullet \\ m(p) + 1 \ , & \textit{if } p \notin \bullet t \textit{ and } p \in t\bullet \\ \quad m(p) \quad , & \textit{else} \end{cases}$$

3 Visual Language Plans

In this section we will introduce and explain the elements of Visual Language Plans. During the Metafora project pedagogues and psychologists of the University of Exeter developed this Visual Language Plans and pedagogues and teachers of the Hebrew University of Jerusalem and the National and Kapodistrian University of Athens evaluated it. There are several pedagogical studies [15,16,17,18,19,20,21,22] using Visual Language Plans, but none of these studies defines a consistent syntax for it. [5] presents an overview of the used definitions of Visual Language Plans during the Metafora project. The unpublished *Guideline for the Visual Language* [23] gives an informal description of the syntax and we will formally define it in this section.

To solve Metafora challenges students build their own Visual Language Plan which describes their approach to the problem. Such a Visual Language Plan consist of nodes and different types of arcs. Nodes of a Visual Language Plan are cards. The meaning of a card depends on their label and this label belongs to the Visual Language. At the moment, the Visual Language has about 60 labels and it allows teachers to add further labels. The Visual Language divides those labels into 7 disjoint categories and this categories belong to different detail levels. We use the categories to define the syntax and semantics of the Visual Language. The 7 categories of the Visual Language are Activity Stage, Gate, Activity Process, Resource, Role, Attitude and Other.

Activity Stage Cards labeled with an Activity Stage model main steps. Some available Activity Stage labels are *explore*, *define questions* and *find hypothesis*.

Gate Cards labeled with a Gate direct the control flow between cards labeled with an Activity Stage. The Visual Language contains gates for *and* and *xor*.

Activity Process Cards labeled with an Activity Process model the actions which students do. Some available Activity Process labels are *simulate*, *discuss* and *make notes*. This cards model the activities required for solving a card labeled with an Activity Stage.

Resource Cards labeled with a Resource label are links to persistent instances of microworlds and tools integrated in Metafora. For example, there are labels for the physics pirate game *PiKI*, the algebraic pattern tool *eXpresser* and for the graphical discussion environment LASAD [10].

Role Cards labeled with a Role annotate required roles for other cards. Some available Role labels are *note taker*, *evaluator* and *manager*.

Attitude Cards labeled with an Attitude annotate required mind-sets. Some available Attitude labels are *rational*, *critical* and *creative*.

Other Cards labeled with Other labels annotate domain specific information. At the moment the only available Other labels are the generic labels *text card* and *blank card*.

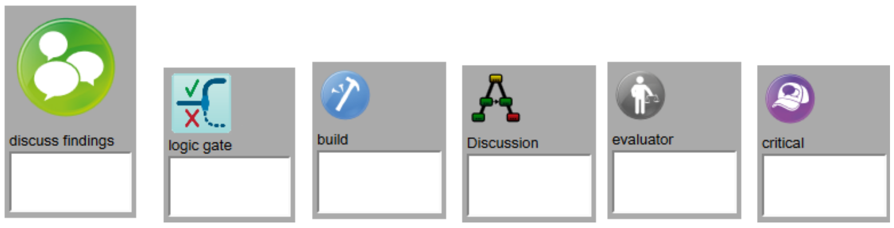

Fig. 2. Examples for cards labeled with labels of the different categories. From the left to the right a card labeled with an *Activity Stage* label, a card labeled with a *Gate* label, a card labeled with an *Activity Process* label, a card labeled with a *Resource* label, a card labeled with a *Role* label and a card labeled with an *Attitude* label.

Figure 2 shows example cards. The category of a cards label is visible through the size, style and coloring of the cards icon. The icons of cards labeled with an *Activity Stage* are larger than other labels to visualize the higher granularity and importance of these cards. To puzzle out the model, the students create labeled card instances and connect it with each other. The students can use 4 different arcs to connect the cards. These arcs are:

is next to The directed *is next to* relation, shown as red arc, models a time sequence of cards. It connects cards labeled with an Activity Stage or a Gate to model the abstract learnflow of a Visual Language Plan. It also orders cards within the set of cards labeled with an Activity Process, within the set of cards labeled with a Resource, within the set of cards labeled with a Role and within the set of cards labeled with an Attitude.

is needed for The directed *is needed for* relation, shown as blue arc, models a time sequence with propagation of resources. It connects cards labeled with an Activity Process, a Role, an Attitude or a Resource with cards labeled with an Activity Process or an Activity Stage. If this relation ends with a card labeled with an Activity Stage the students must finish the other card before they finish the card labeled with an Activity Stage. It also connects cards labeled with a Role or an Attitude with cards labeled with a Role, an Attitude or a Resource and within the set of cards labeled with a Resource.

is input for The directed *is input for* relation, shown as dashed green arc, combines the meaning of the *is next to* and *is needed for* relations. It models propagation of resources to later cards, e.g. for reflection. This relation connects cards labeled with an Activity Process, a Role, an Attitude or a Resource.

is linked to The symmetric *is linked to* relation, shown as black line, models an undirected relation between cards. It associates cards labeled with an Attitude or a Role to cards modeling activities. This relation connects cards labeled with an Activity Process, a Role, an Attitude or a Resource with each other or with cards labeled with an Activity Stage. If it connects a card with a card label with an Activity Stage the students must finish this card before they finish the card labeled with the Activity Stage.

The collaborative web application *Planning Tool* implements the Visual Language and is part of the Metafora system [2]. To solve a challenge students use the Planning Tool and create a Visual Language Plan to model their approach. With the help of their Visual Language Plan the students document their work and access persistent instances of the used mircoworlds and tools. At the moment, we develop a tool which aims to support the students in creating and executing their Visual Language Plan. Therefore, we need a mathematical syntax for Visual Language Plans, based on the given description. To define this syntax, we first define a mathematical structure for Visual Language Plans.

Definition 4 (Visual Language Plan Structure). *A Visual Language Plan consists of a finite set C of labeled cards and 4 relations.*

The set C is the union of the pairwise disjoint sets C_{AS}, C_G, C_{AP}, C_{Re}, C_{Ro}, C_{At} and C_O, where C_{AS} is a set of cards labeled with an Activity Stage, C_G is a set of cards labeled with a Gate, C_{AP} is a set of cards labeled with an Activity Process, C_{Re} is a set of cards labeled with a Resource, C_{Ro} is a set of cards labeled with a Role, C_{At} is a set of cards labeled with an Attitude and C_O is a set of cards labeled with an Other label. Further, the set C_G is the union of the finite disjoint sets $C_{G_{and-split}}$, $C_{G_{and-join}}$, $C_{G_{xor-split}}$ and $C_{G_{xor-join}}$.

The 4 relations are a directed is next to *relation $R_{next} \subseteq ((C_{AS} \cup C_G) \times (C_{AS} \cup C_G)) \cup (C_{AP} \times C_{AP}) \cup (C_{Re} \times C_{Re}) \cup (C_{Ro} \times C_{Ro}) \cup (C_{At} \times C_{At})$, a directed* is needed for *relation $R_{need} \subseteq ((C_{AP} \cup C_{Ro} \cup C_{At} \cup C_{Re}) \times (C_{AS} \cup C_{AP})) \cup ((C_{Ro} \cup C_{At}) \times (C_{Ro} \cup C_{At} \cup C_{Re})) \cup (C_{Re} \times C_{Re})$, a directed* is input for *relation $R_{in} \subseteq ((C_{AP} \cup C_{Ro} \cup C_{At} \cup C_{Re}) \times (C_{AP} \cup C_{Ro} \cup C_{At} \cup C_{Re}))$ and a symmetric* is linked to *relation $R_{link} \subseteq (((C \setminus C_G) \times (C \setminus C_G)) \setminus (C_{AS} \times C_{AS}))$.*

A 4-tuple $(C, R_{next}, R_{need}, R_{in}, R_{link})$ is called Visual Language Plan Structure.

This definition for the syntax of Visual Language Plans is incomplete. It allows modeling splits and joins without using cards labeled with Gates and it does not enforce that a Visual Language Plan is weakly connected. We extend this definition to get a unique behavior of the model, avoid error-prone plans and enable validity checking. Therefore, we need the preset and postset of cards.

Definition 5 (Pre- and Postset, Information Preset). *Given a Visual Language Plan Structure $P = (C, R_{next}, R_{need}, R_{in}, R_{link})$. The preset of a card $c \in C$ is the set of cards $\bullet c := \{c' \in C \mid (c', c) \in (R_{next} \cup R_{need})\}$. The postset of a card $c \in C$ is the set of cards $c\bullet := \{c' \in C \mid (c, c') \in (R_{next} \cup R_{need})\}$*

The information preset $\circ c := \{c' \in C \mid (c', c) \in R_{in}\}$ *of a card $c \in C$ is the preset only considering the* is input for *relation.*

We call a card labeled with an Activity Stage $c \in C_{AS}$ with no other card labeled with an Activity Stage in its preset an *initial card* and a card labeled with an Activity

Stage $c' \in C_{AS}$ with no other card labeled with an Activity Stage in its postset an *end card*. To ensure a unique meaning, we demand that a Visual Language Plan fulfills following properties:

Definition 6 (valid Visual Language Plan Structure). *Given a Visual Language Plan Structure $P = (C, R_{next}, R_{need}, R_{in}, R_{link})$. We call P* valid *if it fulfills all the following properties:*

(I) A visual language plan has one initial card $c_i \in C_{AS}$, i.e. $\bullet c_i \cap C_{AS} = \emptyset$, and one end card $c_e \in C_{AS}$, i.e. $c_e \bullet \cap C_{AS} = \emptyset$. All other cards $c \in C_{AS}$ have one incoming is next to *arc and one outgoing* is next to *arc, i.e. for all other cards $c \in C_{AS} \setminus \{c_i, c_e\}$ exist two unique cards $c', c'' \in C_{AS}$ such that $(c', c) \in R_{next}$ and $(c, c'') \in R_{next}$ hold.*

(II) All cards $c \in C_{G_{and-split}} \cup C_{G_{xor-split}}$ have one incoming is next to *arc and two outgoing* is next to *arcs, i.e. for all $c \in C_{G_{and-split}} \cup C_{G_{xor-split}}$ exist 3 unique cards $c_1, c_2, c_3 \in C_{AS} \cup C_G$, $c_2 \neq c_3$, such that $(c_1, c) \in R_{next}$ and $\{(c, c_2), (c, c_3)\} \subset R_{next}$ hold.*

(III) All cards $c \in C_{G_{and-join}} \cup C_{G_{xor-join}}$ have two incoming is next to *arcs and one outgoing* is next to *arc, i.e. for all $c \in C_{G_{and-join}} \cup C_{G_{xor-join}}$ exist 3 unique cards $c_1, c_2, c_3 \in C_{AS} \cup C_G$, $c_1 \neq c_2$, such that $\{(c_1, c), (c_2, c)\} \subset R_{next}$ and $(c, c_3) \in R_{next}$ hold.*

(IV) The number of and *splits is equal to the number of* and *joins, i.e. $|C_{G_{and-split}}| = |C_{G_{and-join}}|$. The number of* xor *splits is equal to the number of* xor *joins, i.e. $|C_{G_{xor-split}}| = |C_{G_{xor-join}}|$.*

Property (I) ensures that a Visual Language Plan has a unique card labeled with an Activity Stage as start for the execution and all properties together define a very strict structure for the abstract learnflow model. They are implicit contained in the *Guideline for the Visual Language* [23]. This restrictive structure for the low detail cards of a Visual Language Plan supports the students while building their abstract model. The students refine their abstract model with high detail cards afterwards. Property (IV) ensures, together with the other properties, that every *and*-split is joined with an *and*-join and every *xor*-split is joined with an *xor*-join. This is also a requirement given in [23]. Because of the idea of refinement, we have to find for each card with high detail, i.e. each card not labeled with an Activity Stage or Gate, a card labeled with an Activity Stage. Therefore we need paths in a Visual Language Plan.

Definition 7 (Path in a Visual Language Plan). *Given a Visual Language Plan Structure $P = (C, R_{next}, R_{need}, R_{in}, R_{link})$ and a set of arcs $A \subseteq R_{next} \cup R_{need} \cup R_{in} \cup R_{link}$. A* directed path *in P within A from a card $c_1 \in C$ to a card $c_m \in C$ is a sequence $\sigma = c_1, \dots, c_m$ of cards $c_i \in C$ such that for $1 \leq i \leq m-1$: $(c_i, c_{i+1}) \in A$ holds. A* undirected path *in P within A from a card $c_1 \in C$ to a node $c_m \in C$ is a sequence $\sigma = c_1, \dots, c_m$ of cards $c_i \in C$ such that for $1 \leq i \leq m-1$: $(c_i, c_{i+1}) \in A$ or $(c_{i+1}, c_i) \in A$ holds.*

In a Visual Language Plan, a card $c \in C_{AS}$ labeled with an Activity Stage is usually refined with the help of other cards. Those other cards are direct or indirect connected to

c with *is needed for* or *is linked to* arcs. We call a card which refines a card labeled with an Activity Stage a subordinated card. From the idea of refinement of cards follows that a card can only be subordinate to one card labeled with an Activity Stage.

Definition 8 (Subordination). *Given a valid Visual Language Plan Structure $P = (C, R_{next}, R_{need}, R_{in}, R_{link})$ and a card labeled with an Activity Stage $c \in C_{AS}$. We call the set of nodes $S_{1,c} := (\bullet c \setminus C_{AS}) \cup \{c^* \in C \mid (c^*, c) \in R_{link}\}$ the* set of first order subordinated cards to *c. A card $c' \in C$ is subordinated to c if a card $c'' \in S_{1,a}$ and a undirected path σ within $R_{next} \cup R_{need} \cup R_{link}$ from c' to c'' exist such that σ does not contain the card c. S_c is the set of all subordinated nodes of c.*

While executing a plan, students often need achievements from earlier stages to solve later stages. They can use the *is input for* relation to propagate a resource to a later card. This requires that they finished the earlier task before they can start the later task. Further, we demand that the connected cards refine different cards labeled with an Activity Stage.

Definition 9 (Visual Language Plan). *Given a valid Visual Language Plan Structure $P = (C, R_{next}, R_{need}, R_{in}, R_{link})$. We call P a* Visual Language Plan*, if it fulfills all the following properties:*

(V) P is weakly connected, i.e. for each two cards $c, c' \in C$ exist an undirected path within $R_{next} \cup R_{need} \cup R_{in} \cup R_{link}$ from c to c''.

(VI) Each card not labeled with an Activity Stage or a Gate is subordinated to precisely one card labeled with an Activity Stage, i.e. for all cards $c \in C \setminus (C_{AS} \cup C_G)$ a unique card $c' \in C_{AS}$ exist such that $c \in S_{c'}$ holds.

(VII) Each is input for *arc connects cards which are subordinated to different cards labeled with an Activity Stage, i.e. for all $(c_1, c_2) \in R_{in}$, $c_1, c_2 \in C$, unique different cards $c', c'' \in C_{AS}$ exist such that $c_1 \in S_{c'}$ and $c_2 \in S_{c''}$ holds.*

(VIII) The is next to, is needed for *and* is linked to *relations only connect cards not labeled with an Activity Stage which are subordinated to the same card labeled with an Activity Stage, i.e. for each pair of cards $c_1, c_2 \in C \setminus (C_{AS} \cup G)$ with $(c_1, c_2) \in R_{next} \cup R_{need} \cup R_{link}$ exist a card $c \in C_{AS}$ such that $c_1 \in S_c$ and $c_2 \in S_c$ holds.*

4 Semantics of the visual language

In this section we will define the semantics of a Visual Language Plan with the help of Petri nets. A card is in one of 3 states *idle*, *started* or *done*. A card shows its state with its coloring. The coloring of an idle card is grey, the coloring of a started card is yellow and the coloring of a finished card is green. Through this coloring, the students are able to see what they did and what is next. This is important because most of the Metafora challenges need more than 4 school lessons and include homework sessions. The set of the state of all cards is the state of the plan. This state function maps the set of cards C to $\{1, 2, 3\}$. With this notation 1 means idle, 2 means started and 3 means finished.

Definition 10 (State of a Visual Language Plan). *Given a Visual Language Plan $P = (C,\ R_{next},\ R_{need},\ R_{in},\ R_{link})$. The state of P is a function $s : C \rightarrow \{1, 2, 3\}$.*

We call a card $c \in C$ idle *iff* $s(c) = 1$, started *iff* $s(c) = 2$ *and* finished *iff* $s(c) = 3$. *We call a Visual Language Plan finished, if its end card is finished.*

In the *Planning Tool*, a card change its coloring if a student select this card as started or finished. If a student select a card labeled with a Resource as started the linked tool opens in a new Metafora tab. Still, Metafora does not clearly define the semantics of a Visual Language Plan. The *Planning Tool* allows students to mark cards as started or finished without checking any rules. At the moment, the semantics for Visual Language Plans is only given as informal description [23]. To develop a Metafora workflow engine, we need to analyze state changes of cards. Therefore, we require the formal semantics of Visual Language Plans. We extracted the following execution rules from the informal descriptions:

(a) Students must start a card before they can finish it.
(b) Students only can start a card if they have started all cards before that card.
(c) Students can finish a card if all they have finished all cards before that card.
(d) Students can only choose one path after a xor-gate.
(e) Students must solve both paths after an and-gate before they can finish the joining and-gate.
(g) Students must start a card labeled with an Activity Stage before they start their refining cards.
(h) Students must finish all refining cards of a card labeled with an Activity Stage before they can finish the card labeled with an Activity Stage.
(i) For cards connected with the *is input for* relation, students can only start the successor if they finished the predecessor before.

These rules overlap and interfere. For example there is a clash of rule (b) and (d) for the joining card of an xor-split. The meaning of *before* in rule (b) and (c) is different. Through the different detail level of cards it depends on their neighborhood if they can change their state. The occurrence rule for cards of a Visual Language Plan is a complicated logical formula which is expensive to test. To avoid this, we decided to define the semantics of a Visual Language Plan with the help of a Petri net. For our mapping, we consider the starting and finishing of cards as events. In the Petri net to a Visual Language Plan transitions represent this events. If we only consider the starting of cards, the Petri net roughly looks like the Visual Language Plan. If we only consider the finishing of cards, the Petri net roughly looks like the Visual Language Plan, too. Places and arcs which control the learnflow connect these parts. In the following we will give step by step a mapping for a Visual Language Plan $P = (C, R_{next}, R_{need}, R_{in}, R_{link})$ to a Petri net $N = (S, T, F, m_0)$. The Petri net N defines the semantics of the plan P. The following steps lead to the corresponding Petri net N:

Step 1: Choose a fixed enumeration for all cards $c \in C$, i.e. a bijection $M : C \to \mathbb{N}_0$, and add for each card $c \in C$ two transitions $t_{M(c),s}$ and $t_{M(c),f}$ to the net N. For a card $c \in C$, the transition $t_{M(c),s}$ represents the starting of c and the transition $t_{M(c),f}$ represents the finishing of c. To make sure that the finishing event of a card can only occur after the starting event, add for each card $c \in C$ a place $p_{M(c)}$ between $t_{M(c),s}$ and $t_{M(c),f}$, i.e. add a place $p_{M(c)}$ and the two edges $(t_{M(c),s},\ p_{M(c)})$ and $(p_{M(c)},\ t_{M(c),f})$ to N. All places $p_{M(c)}$ are not marked, i.e. $\forall p_{M(c)} \in P\ :\ m_0(p_{M(c)}) = 0$.

Figure 4 shows our enumeration for the example and Figure 3 shows the mapping of a card $c \in C$ to their event transitions.

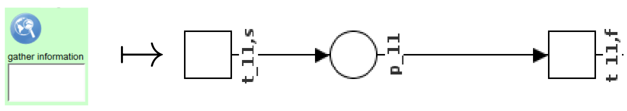

Fig. 3. Step 1: We map the card c with $M(c) = 11$ on two transitions $t_{11,s}$ and $t_{11,f}$ representing their start and finish events and add a places p_{11}.

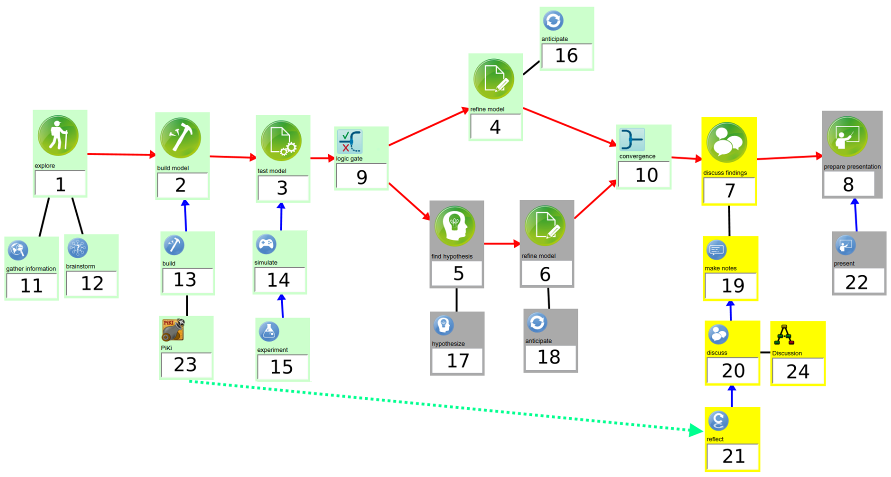

Fig. 4. This figure shows the bijection $M : C \to \mathbb{N}_0$ which we use for our example. For each card $c \in C$ we wrote the value $M(c)$ on that card.

Step 2: To transfer the *is next to* relation, add for each arc $(c, c') \in R_{next}$ a new place $p_{next,M(c),M(c'),s}$ between $t_{M(c),s}$ and $t_{M(c'),s}$, i.e. add $p_{next,M(c),M(c'),s}$ and the edges $(t_{M(c),s}, p_{next,M(c),M(c'),s})$ and $(p_{next,M(c),M(c'),s}, t_{M(c'),s})$ to N. This ensures that the students can only start the succeeding card c' if they started the preceding card c before. Further, add for each arc $(c, c') \in R_{next}$ a place $p_{next,M(c),M(c'),f}$ between $t_{M(c),f}$ and $t_{M(c'),f}$ to the net, i.e. add $p_{next,M(c),M(c'),f}$ and the edges $(t_{M(c),f}, p_{next,M(c),M(c'),f})$ and $(p_{next,M(c),M(c'),f}, t_{M(c'),f})$ to N. This ensures that the finish event of the succeeding card can only occur if the finish event of the preceding card already occurred.

Figure 5 shows the mapping of the *is next to* relation between cards labeled with an Activity Stage.

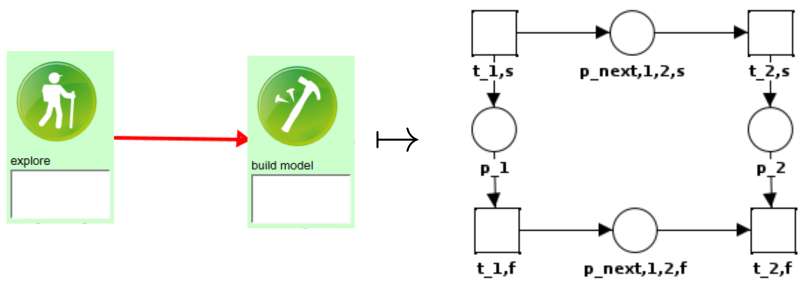

Fig. 5. Step 2: This figure shows how we map the *is next to* relation between the card c labeled with *exlpore*, $M(c) = 1$, and the card c' labeled with *build model*, $M(c') = 2$, to the Petri net. The place $p_{next,1,2,s}$ make sure that c starts before c'. The place $p_{next,1,2,f}$ make sure that c finish before c'.

Step 3: If students use the *is next to* relation with an *XOR* split, they can only start one successor card because of rule (d). To fulfill this rule we add for each card $c \in C_{G_{xor-join}}$ a place $p_{xor,M(c),s}$ between the $t_{M(c),s}$ and the starting event transitions for all cards labeled with an Activity Stage or a Gate in the postset of c, i.e. add $p_{xor,M(c),s}$, $(t_{M(c),s},\ p_{xor,M(c),s})$ and for each card $c' \in c\bullet \cap (C_{AS} \cup C_G)$ an edge $(p_{xor,M(c),s}, t_{M(c'),s})$ to N. Through $p_{xor,M(c),s}$ the places $\{p_{next,M(c),M(c'),s} \mid c' \in c\bullet \cap (C_{AS} \cup C_G)\}$ are superfluous and we removed it. If we would only want to fulfill rule (d) we could stop now but we want to keep the symmetry of the Petri net and avoid useless marked places. Add $p_{xor,M(c),f}$, $(t_{M(c),s},\ p_{xor,M(c),f})$ and for each card $c' \in c\bullet \cap (C_{AS} \cup C_G)$ an edge $(p_{xor,M(c),f},\ t_{M(c'),s})$ to N.

Figure 6 shows the result of this mapping of an xor-split.

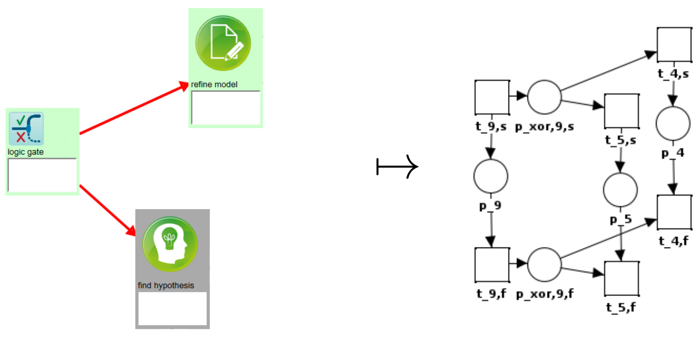

Fig. 6. Step 3: This figure shows the result of the mapping of an xor-split between the cards c labeled with *logic gate*, $M(c) = 9$, the card c' labeled with *refine model*, $M(c') = 4$, and the card c'' labeled with *find hypothesis*, $M(c'') = 5$, to the Petri net.

Step 4: If students use the *is next to* relation with a *XOR* join, they can choose only one path. Due to property (III) there are two preceding cards before this *XOR* join. Because of those two preceding cards, we added two places to N in Step 1. The students can only execute one path before this *XOR* join. This cause a deadlock. To solve this problem melt those two places into one place. For each card $c \in C_{G_{xor-join}}$ remove all places $\{p_{next,M(c'),M(c),s} \mid c' \in \bullet c \cap (C_{AS} \cup C_G)\}$ and add a place $p_{xor,M(c),s}$, an arc $(p_{xor,M(c),s}, t_{M(c),s})$ and arcs $\{(t_{M(c'),s},\ p_{xor,M(c),s} \mid c' \in \bullet c \cap (C_{AS} \cup C_G)\}$. We also do this for the finish event part of our net. For each card $c \in C_{G_{xor-join}}$ remove all places $\{p_{next,M(c'),M(c),f} \mid c' \in \bullet c \cap (C_{AS} \cup C_G)\}$ and add a place $p_{xor,M(c),f}$, an arc $(p_{xor,M(c),f}, t_{M(c),f})$ and arcs $\{(t_{M(c'),f},\ p_{xor,M(c),f} \mid c' \in \bullet c \cap (C_{AS} \cup C_G)\}$.

Figure 7 shows the result of this mapping of an xor-join.

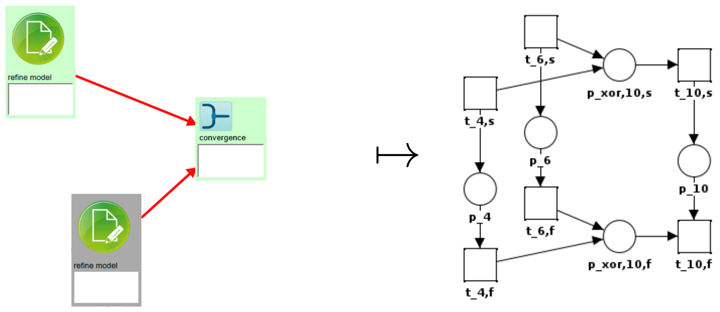

Fig. 7. Step 4: This figure shows the result of the mapping of an xor-join between the cards c labeled with *convergence*, $M(c) = 10$, and the cards c' labeled with *refine model*, $M(c') = 4$, and c'' labeled with *refine model*, $M(c'') = 6$.

Step 5: To transfer the *is needed for* relation, add for each arc $(c, c') \in R_{need}$ a new place $p_{need,M(c),M(c'),s}$ between $t_{M(c),s}$ and $t_{M(c'),s}$, i.e. add $p_{need,M(c),M(c'),s}$ and the edges $(t_{M(c),s}, p_{need,M(c),M(c'),s})$ and $(p_{need,M(c),M(c'),s}, t_{M(c'),s})$ to N. This ensures that students can only start the succeeding card c' after they started the preceding card c. Further, add for each arc $(c, c') \in R_{need}$ a place $p_{need,M(c),M(c'),f}$ between $t_{M(c),f}$ and $t_{M(c'),f}$ to the net, i.e. add $p_{need,M(c),M(c'),f}$ and the edges $(t_{M(c),f}, p_{need,M(c),M(c'),f})$ and $(p_{need,M(c),M(c'),f},\ t_{M(c'),f})$ to N. This ensures that the finish event of the succeeding card can only occur if the finish event of the preceding card already occurred.

This mapping is similar to the mapping of the *is next to* relation in Step 1.

Step 6: If students use the *is needed for* relation to model subordination, the places added with the last step enforce that the starting events of the subordinated cards occur before the starting event of the card labeled with an Activity Stage can occur. This is wrong and we remove those places, i.e. for each card $c \in C_{AS}$ remove all places $\{p_{need,M(c'),M(c),s} \mid c' \in S_c\}$. For each card $c \in C_{AS}$, all to c subordinated cards are only allowed to start after c. We enforce this by adding places between c and all to c subordinated cards, i.e. for each $c \in C_{AS}$ and each $c' \in S_c$ add a place $p_{sub,M(c),M(c'),s}$

and edges $(t_{M(c),s}, p_{sub,M(c),M(c'),s})$ and $(p_{sub,M(c),M(c'),s}, t_{M(c'),s})$. Rule (h) raise the requirement that for each card $c \in C_{AS}$ all subordinated cards finish before the finish event of c can occur. We make this sure by adding places between the to c subordinated cards and c, i.e. for each $c \in C_{AS}$ and each $c' \in S_c$ add a place $p_{sub,M(c),M(c'),f}$ and edges $(p_{sub,M(c),M(c'),f}, t_{M(c),f})$ and $(t_{M(c'),f}, p_{sub,M(c),M(c'),f})$.

Figure 8 shows the result of this step.

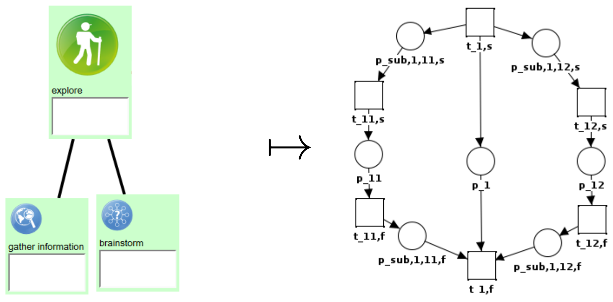

Fig. 8. Step 6: This figure shows the mapping of subordination of the cards c' labeled with *gather information*, $M(c) = 11$, and c'' labeled with *brainstorm*, $M(c'') = 12$, to the card c labeled with *explore*, $M(c) = 1$.

Step 7: To transfer the *is input for* relation, add for each arc $(c, c') \in R_{in}$ a new place $p_{in,M(c),M(c')}$ between $t_{M(c),f}$ and $t_{M(c'),s}$, i.e. add $p_{in,M(c),M(c')}$ and the edges $(t_{M(c),f}, p_{in,M(c),M(c')})$ and $(p_{in,M(c),M(c')}, t_{M(c'),s})$ to N. This ensures that the succeeding card c' can only start after finishing the preceding card c.

Figure 9 shows the mapping of the *is input for* relation.

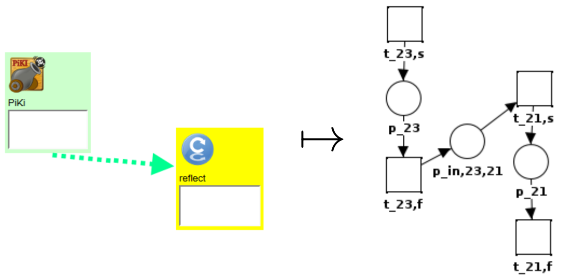

Fig. 9. Step 7: This figure shows how we map the *is input for* relation between the card c labeled with *PiKI*, $M(c) = 23$, and the card c' labeled with *reflect*, $M(c') = 21$, to the Petri net. The place $p_{in,23,21}$ make sure that the students finished c before they can start c'.

Step 8: Finally, two convenience places $p_{initial}$ and p_{end} are added. For the unique initial card c_i add a place $p_{initial}$ and an edge $(p_{initial}, t_{M(c_i),s})$ with $m_0(p_{initial}) = 1$.

The marking of the initial place $p_{initial}$ is 1 if the students did not start executing the Visual Language Plan P. For the unique end card c_e add a place p_{end} and an edge $(t_{M(c_e),f}, p_{end},)$ with $m_0(p_{initial}) = 0$. The marking of this place is 1 if the students finished the Visual Language Plan P.

The *is linked to* relation does not restrict starting or finishing of cards and we do not need to translate it. The language L of this Petri net N is the language of P, i.e. for each transition sequence $\sigma \in L$ starting and finishing of cards according to this sequence is valid for P with respect to the rules given above.

For our example Visual Language Plan we have chosen the mapping shown in Figure 4. Figure 10 shows the Petri net corresponding to this Visual Language Plan which results from the mapping described above. The upper part of the net consists of the transitions which control the starting sequence of the cards labeled with an Activity Stage and has a similar structure as the cards labeled with an Activity Stage in our example. The starting of a card labeled with an Activity Stage enables the start event transitions of their subordinated cards. The Petri net has more places than required and an algorithm for deletion of implicit places could remove $p_{sub,3,14,s}$. It is difficult to decide if a place is an implicit place and we need a fast mapping from the visual language plan to the Petri net so we keep those places. The middle part of the Petri net models the grey, yellow and green sequence. The place $p_{in,23,21}$ for the *is input for* relation from the *PiKI* card to the *reflect* card is also in the middle part. This is the only connection form the lower part of the Petri net to its upper part. The finishing event of the refined cards can only occur after the finishing event of all their refining cards.

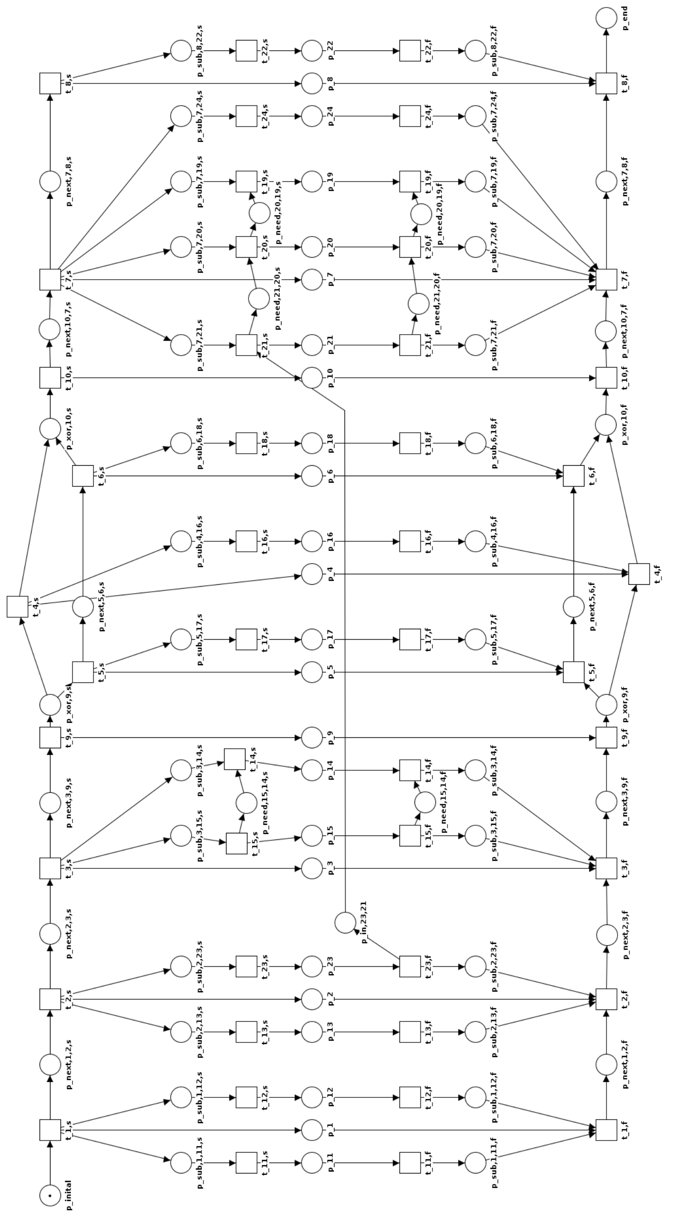

Fig. 10. Petri net for our example. The upper and lower section contain the transitions for the yellow and green occurrence of the activity stage cards. The middle section comprises the transitions for the refining cards.

5 Applications of the Visual Language

With the help of the formal syntax, shown in Section 3, we can automatically check if a Visual Language Plan is valid while students are modeling it. The Metafora learnflow engine we develop will listen to the logs of the *Planning Tool*, analyze the events done by the students and send feedback messages to the students. It will send affirmative feedback messages if the students fulfill desired properties, like refining cards labeled with an Activity Stage. If the students violate syntactic rules it will send corrective feedback messages. For example, this is the case if students connect cards labeled with an Activity Stage with an *is linked to* relation. The feedback messages help the students to create a meaningful and selfregulated learnflow model. Besides all this, we need the formal definition of a syntax for Visual Language Plans to define a formal semantics.

The Petri net mapping for a Visual Language Plan, shown in Section 4, defines a formal semantics for Visual Language Plans. The Metafora learnflow engine will create the Petri net for each Visual Language Plan and use it to analyze the starting and finishing events, done by students to generate helpful feedback messages. The feedback messages are affirmative, corrective or informative. The learnflow engine send an affirmative feedback message if students respect the execution rules, e.g. if a student finished the card labeled with the Activity Stage *explore*. If students violate the execution order of the Visual Language Plan it sends corrective feedback messages. For example, if students start the card labeled with the Activity Stage *test model* before they start the card labeled with the Activity Stage *build model* all students will get a corrective feedback message telling them to build the model first. We use informative feedback messages to tell students working on the same Visual Language Plan about meaningful actions. If Bob and Alice work on the same Visual Language Plan the learnflow engine will send Alice the informative feedback message *'Bob finished build model.'* when Bob changes the state of the card labeled with the Activity Stage *build model* to finished. With these feedback messages we intend to help the students planning and executing their learnflow. We want to shorten the training phase and help the students to concentrate on their learnflow instead of think about occurrence rules for cards. With the help of informative feedback messages we try to help the students keeping track of current state of their learnflow while they use microworlds.

In Metafora, our learnflow engine is not able to enforce the syntax or semantics of a Visual Language Plan. Furthermore, the pedagogical case studies of the Metafora project showed that the students often use their Visual Language Plan for reflecting about their actions and rearrange specific elements to document how the learning actually took place. Reflection is one of the L2L2 behaviors which we support to grant more flexibility on the students side and enable a tight engagement in the planning and execution phases. To do this, we have to change a learnflow during the execution and transfer a state from a Visual Language Plan to its Petri net. The changing of the learnflow or faulty starting and finishing of cards can cause an invalid state of the Visual Language Plan. In case of a faulty state of the Visual Language Plan, a direct mapping would cause a not reachable marking of the Petri net and result in unwanted behavior.

In the following we will describe an approach to transfer a state from a Visual Language Plan to a marking of the corresponding Petri net which can handle faulty states and calculates valuable information to generate useful feedback. In case of a faulty state change of a card or a change of the learnflow model, we collect the state change events which caused the current state of the Visual Language Plan. This means for a Visual Language Plan $P = (C, R_{next}, R_{need}, R_{in}, R_{link})$ with its state $s : C \rightarrow \{1, 2, 3\}$ and the corresponding Petri net $N = (S, T, F, m_0)$ we calculate a set E of transitions which occurred to reach this state. We do this by checking the state of each card $c \in C$ and get the transition set $E = \{t_{M(c),s} \in T \mid c \in C \ \wedge \ s(c) \geq 1\} \cup \{t_{M(c),f} \in T \mid c \in C \ \wedge \ s(c) = 2\}$. Next, we calculate for the net N a valid sequence σ of the transitions contained in E by occurring all enabled transitions, adding them to σ and removing them from E. We do this iterative until E is empty or has no more enabled transitions. For Visual Language Plans, we can do this because we have the state information for joining and splitting cards no conflicts can happen. Now, we have a maximal valid sequence σ of transitions of E, a subset $E' \subseteq E$ of faulty transitions and can easily get a set $A \subseteq T$ of enabled transitions. With this information we can tell the students about the cards with faulty states by analyzing E'. We can recommend possible cards to the students by analyzing A. Moreover, we can calculate a minimal sequence σ', with prefix σ, which enable all transitions $t \in E'$ and recommend steps leading to a valid state of the Visual Language Plan with the help of σ'.

Figure 11 shows our example plan with annotations for the current state of the plan and Figure 12 shows the corresponding Petri net to this plan with a marking corresponding to the current state.

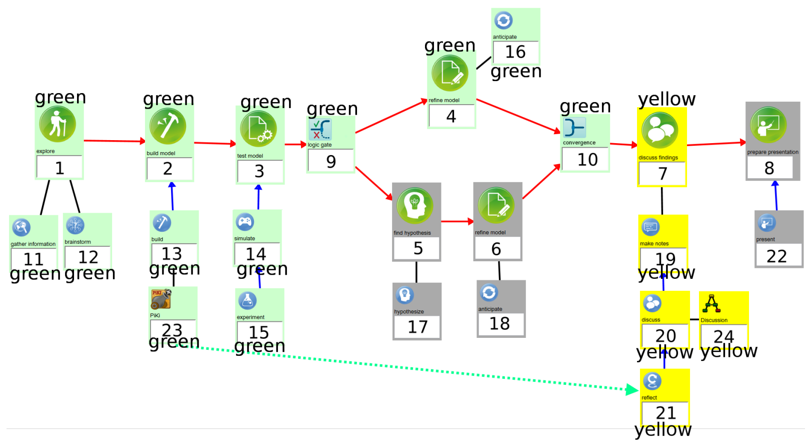

Fig. 11. This figure shows our example Visual Language Plan. For each card $c \in C$ we wrote value $M(c)$ on that card. The cards' states are visible trough the coloring of the cards and we annotated the card with their color for print versions.

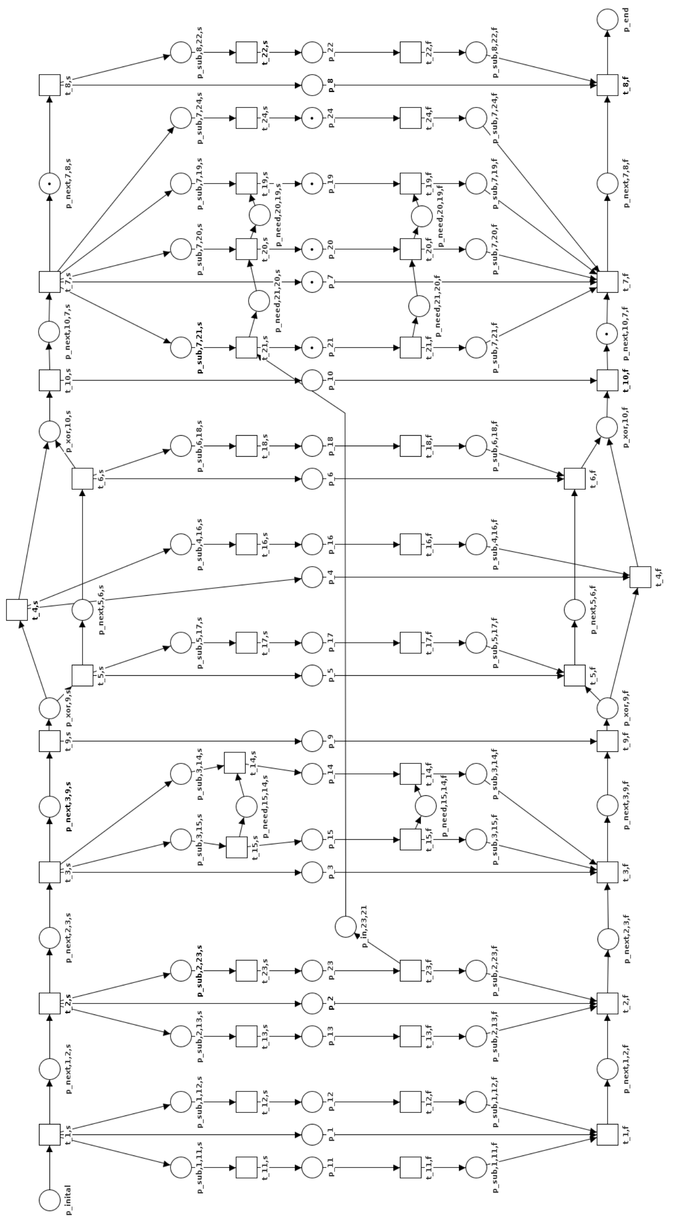
Fig. 12. This figure shows the Petri net for our example shown in Figure 11 with the marking for the state of the plan.

If a student starts the card c labeled with *present* ($M(c) = 22$) the Metafora learnflow engine evaluates this event as occurrence of the not enabled transition $t_{22,s}$. Because of this faulty state change, it calculates the sequence σ of transitions which cause the marking of Figure 12, the set $E = \{t_{22,s}\}$ and the set $A = \{t_{21,f}, t_{20,f}, t_{19,f}, t_{24,f}, t_{8,s}\}$. Now, it unfolds the Petri net with this marking and find the minimal sequence $\sigma' = \sigma, t_{8,s}$. σ' enable all transitions in $E = \{t_{22,s}\}$. Finally, the learnflow engine sends a feedback message recommending to start the card labeled with *prepare presentation*.

6 Conclusion

The Metafora project developed the Visual Language Plans for learnflow modelling. Visual Language Plans support the pedagogy of L2L2. Metaora is a web-based computer supported collaborative learning platform implementing this Visual Language Plans and using Web 2.0 features. The Metafora project did not develop a formal syntax or semantics for this Visual Language Plans.

We develop a Metafora learnflow engine for automatic support of students using the Metafora system. In this paper we give an overview of Visual Language Plans for modeling learnflows. Further, we extracted consistent rules for the syntax and semantics of Visual Language Plans from the available publications and developed a formal syntax and semantics for this plans. To define the semantics of Visual Language Plan we presented a mapping to a Petri net.

The Metafora learnflow engine will analyze events done by students and support them while planning and executing Visual Language Plans. With the syntax for Visual Language Plans the Metafora learnflow engine can generate feedback messages supporting the students in modeling their Visual Language Plan and with the semantics it can generate feedback messages supporting the students while editing and executing their plan. This feedback messages are affirmative, corrective or informative [12]. Furthermore, the corresponding Petri net for a Visual Language Plan enables the Metafora learnflow engine to recommend steps to the students for reaching a valid state.

References

1. Metafora: Project website. `http://www.metafora-project.org/`
2. Metafora: Demo System. `https://www.metafora-project.de/`
3. Metafora Glossary: Visual Language. `http://static.metafora-project.de/VisualLanguage.html`
4. Metafora Glossary: Learning To Learn Together. `http://static.metafora-project.de/L2L2.html`
5. Yang, Y., Wegerif, R., Dragon, T., Mavrikis, M., McLaren, B.M.: Learning how to learn together (L2L2): Developing tools to support an essential complex competence for the internet age. In Rummel, N., Kapur, M., Nathan, M., Puntambekar, S., eds.: CSCL 2013 Conference Proceedings. Volume 2., Madison, International Society of the Learning Sciences (2013) 193–196
6. Dragon, T., Mavrikis, M., McLaren, B.M., Harrer, A., Kynigos, C., Wegerif, R., Yang, Y.: Metafora: A web-based platform for learning to learn together in science and mathematics. Learning Technologies, IEEE Transactions on **6**(3) (2013) 197–207

7. Mavrikis, M., Dragon, T., Yiannoutsou, N., McLaren, B.M.: Towards Supporting 'Learning To Learn Together' in the Metafora platform. Presented at the Intelligent Support for Learning in Groups workshop at the 16th International Conference on Artificial Intelligence in Education (AIED 2013) (2013)
8. Abdu, R., Schwarz, B.: "Metafora" and the fostering of collaborative mathematical problem solving. `http://www.academia.edu/1784715/_Metafora_and_the_fostering_of_collaborative_mathematical_problem_solving`
9. Metafora: Challenge: The Bouncing Cannon Ball. `http://static.metafora-project.de/BouncingCannonBall.html`
10. Humboldt Universität Berlin: LASAD. `http://cses.informatik.hu-berlin.de/research/details/lasad/`
11. Metafora: Report D 2.1 - Visual Language for Learning Processes. `http://data.metafora-project.de/reportD2_1.pdf`
12. Harrer, A., Pfahler, K., Lingnau, A.: Planning for Life-Educate Students to Plan: Syntactic and Semantic Support of Planning Activities with a Visual Language. In Chen, N.S., Huang, R., Kinshuk, Li, Y., Sampson, D.G., eds.: Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on, Washington, IEEE, CPS (2013) 309–313
13. Bergenthum, R., Desel, J., Harrer, A., Mauser, S.: Learnflow mining. In Seehusen, S., Lucke, U., Fischer, S., eds.: DeLFI 2008. Volume P-132 of LNI., Bonn, Geselschaft für Informatik (2008) 269–280
14. Petri, C.A.: Kommunikation mit Automaten. Dissertation, Technische Hochschule Darmstadt (1962)
15. Smyrnaiou, Z., Moustaki, F., Yiannoutsou, N., Kynigos, C.: Interweaving meaning generation in science with learning to learn together processes using Web 2.0 tools. Themes in Science and Technology Education **5**(1-2) (2013) 27–44
16. Kynigos, C., Moustaki, F.: Designing tools to support group work skills for constructionist mathematical meaning generation. `http://data.metafora-project.de/P12_kynigos_moustaki.pdf` (2013)
17. Daskolia, M., Yiannoutsou, N., Xenos, M., Kynigos, C.: Exploring Learning-to-learn-together Processes within the Context of an Environmental Education Activity. In: Proceedings of The Ireland International Conference on Education - IICE-2012. (2012)
18. Abdu, R., DeGroot, R., Drachman, R.: Teacher's Role in Computer Supported Collaborative Learning. In: CHAIS conference. (2012) 1–6
19. Smyrnaiou, Z., Varypari, E., Tsouma, E.: Dialogical interactions concerning the scientific content through face to face and distance communication using web 2 tools. In Pintó, R., López, V., Simarro, C., eds.: Computer Based Learning in Science Conference Proceedings 2012, Barcelona, CRECIM (2012) 117–125
20. Pifarré, M., Wegerif, R., Guiral, A., del Barrio, M.: Developing Technological and Pedagogical Affordances to Support Collaborative Inquiry Science Processes. In Demetrios, S.G., Spector, M.J., Ifenthaler, D., Isaias, P., eds.: IADIS International Conference on Cognition and Exploratory Learning in Digital Age, Lisbon, IADIS Press (2012) 139–147
21. Moustaki, F., Kynigos, C.: Meanings for 3d mathematics shaped by online group discussion. In Kynigos, C., Clayson, J.E., Yiannoutsou, N., eds.: Constructionism 2012 Conference - Theory, Practice and Impact, Athens, The Educational Technology Lab (2012) 174–183
22. Yiannoutsou, N., Kynigos, C.: Boundary Objects in Educational Design Research: designing an intervention for learning how to learn in collectives with technologies that support collaboration and exploratory learning. In: Educational design research - Part B: Illustrative cases. SLO, Enschede (2013) 357–381
23. Pfahler, K.: Guideline for the Visual Language. `http://data.metafora-project.de/VisualLanguageGuideline.pdf`