Framework for an Interactive Assistance in Diagnostic Processes Based on the Translation of UML Activities into Petri Nets

Patrick Philipp*, Yvonne Fischer†, Dirk Hempel‡ and Jürgen Beyerer*†
*Vision and Fusion Laboratory IES, Karlsruhe Institute of Technology KIT, Karlsruhe, Germany
Email: p.philipp@kit.edu
†Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB, Karlsruhe, Germany
Email: {yvonne.fischer, juergen.beyerer}@iosb.fraunhofer.de
‡Steinbeis-Transfer-Institut Klinische Hämatoonkologie, Donauwörth, Germany
Email: dirk.hempel@gmail.com

***Abstract*—In contemporary medicine, diagnostic processes provided by Clinical Practice Guidelines (CPGs) play a key role. A CPG provides recommendations that open up a scope of actions for the medical practitioner. But there is a gap between theoretical knowledge and practical solutions. Furthermore, barriers of implementation can arise. Taking these challenges into account, we propose a framework for facilitating the implementation of diagnostic processes. For this reason, the CPGs of Chronic Myeloid Leukemia (CML) and Myelodysplastic Syndromes (MDS) are modeled using Unified Modeling Language (UML). A UML activity serves as a basis for more complex models that are used to provide the actual assistance functions. This paper focuses on the automatic translation of UML activities into Petri nets.**

Keywords: Clinical Practice Guidelines, Diagnosis, Assistance

I. Introduction

The huge amount of publications can be seen as a challenge in the medical diagnostic today: a search of the term "chronic myeloid leukemia diagnosis" in Pubmed [1] yields about 17000 results. Consequently, a profound inquiry concerning a specific issue can be extremely laborious. Clinical Practice Guidelines (CPGs) offer a possible solution to prevent drowning in information. That is because they are able to provide consolidated medical knowledge condensed into general recommendations for actions [2].

To implement a CPG, the medical practitioner has to adapt the recommendations to the given boundary conditions (e.g. patient's wishes, medical equipment) [3], [4]. Consequently, there is a gap between theoretical knowledge on the one side and practical solutions on the other side [5]. In addition, barriers can arise if a guideline is not fully accepted by the expert. This can be the consequence of (for example) a fear of regimentation or because the guideline does not reflect the latest developments in the field [6].

To support the medical practitioner during the implementation of a diagnostic process, we propose an interactive assistance that helps to reduce the gap between theoretical knowledge and practical solutions and helps to lower barriers.

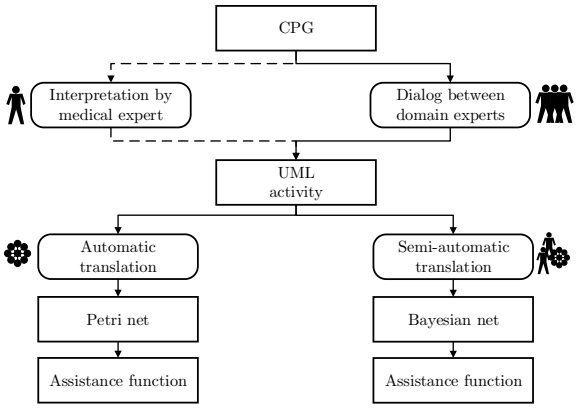

Figure 1. A CPG can be transformed into a UML activity by an expert's dialog or by the medical expert himself. An activity serves as an interface to the models used for providing the actual assistance functions [2], [5].

II. Modeling Approach

To establish a CPG model represented by a UML activity, the medical knowledge provided by the guideline (e.g. running texts, diagrams) has to be formalized. We believe that this can be done via a dialog of experts from the medical- as well as the technical domain (see Figure 1). Alternatively, the medical expert can interpret the guideline on his own. Moreover, the depicted bypass enables a medical practitioner to modify an already existing UML activity by himself. We are convinced that by this, most of the barriers of CPG implementation can be moderated. This includes for example the reduction of fear of regimentation or short-term modifications due to medical symposia.

To provide the actual assistance functions, (i.e. propose suitable examination values to the practitioner during the diagnostic process), an activity is translated into more complex models. These models can be (semi-) automatically generated by only one given activity. In [2], [5] we introduced the translation from UML activities into Bayesian nets (see Figure 1). In this paper we focus on the translation of UML activities into Petri nets. It is based on the work of Störrle et. al [7], [8], [9].

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.110

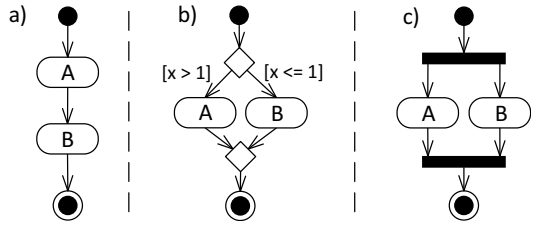

Figure 2. Three typical routings of the CPG models of CML and MDS.

III. UML ACTIVITIES

UML is accepted in software industry worldwide [10]. Activities have been chosen as an interface because of their easy comprehensibility for the medical- as well as the technical domain experts [2]. This is a necessary precondition to make the dialog of experts work smoothly and to allow modifications by the medical practitioner on his own.

A. Fundamentals

An activity answers the question of how a particular process or algorithm proceeds [11]. Control flows, object flows, actions, decisions and forks can be used to specify such an activity [10]. Figure 2 shows the typical routings which appear in the CPG models of Chronic Myeloid Leukemia (CML) and Myelodysplastic Syndromes (MDS) [2]. The black dot represents the start of an activity (initial node), whereas the double circle corresponds to the end of an activity (activity final). The rounded rectangles are the actions that are to be performed, while the arrows represent the flow. In Subfigure a) the actions are carried out one after another (sequentially). Subfigure b) depicts a selective routing. I.e., only one of the two actions A and B is performed. The corresponding decision is represented by a diamond (decision node). In this example the decision depends on a variable x which is either greater 1 or not. The second diamond is called merge node as it merges both flows. Subfigure c) shows a case where two actions can be performed concurrently. The flow is split up by a fork node (black bar). The join node on the right synchronizes the flows. As a result, the flow continues only iff both actions have been performed in any arbitrary order.

B. Modeling of a Particular Disease

Figure 3 depicts the UML activity based on the CPG of CML. Due to its size, special parts of interest are emphasized by magnifying glasses. Subfigure a) shows three sequential actions: “suspicion of CML”, “anamnesis” and “physical examination”. The first action “suspicion of CML” is a necessary precondition for the appliance of the diagnosis algorithm of the CPG for CML. During the second action “anamnesis” the patient is asked if he experiences bone pain. We used a so called pin notation to specify that the value “bone pain” is an output parameter generated by the action “anamnesis”. The third action “physical examination”

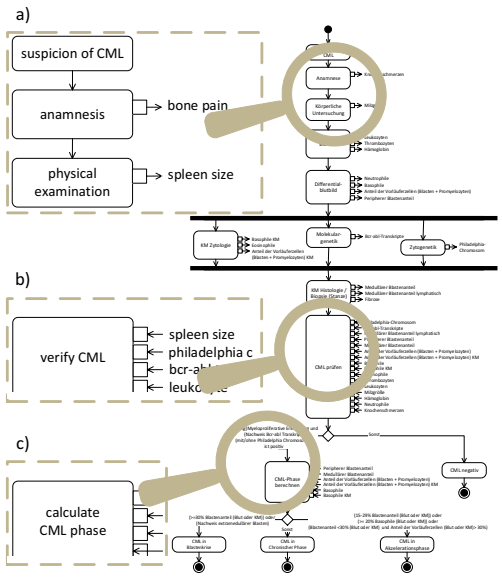

Figure 3. In the background of this figure the UML activity for CML is shown. Because of the size of the activity, the sketch emphasizes some parts of interest (magnifying glasses) [2], [5].

yields the output parameter “spleen size”. In general, actions can have several output or input parameters. Subfigure b) depicts the action “verify CML”. This action involves an assessment by the medical expert: He has to decide whether or not the gathered examination values indicate the presence of the disease or not. This decision cannot be modeled in a deterministic way (i.e. not by fixed rules), because the final decision is up to the medical expert. Subfigure c) shows another type of decision which can be made on the basis of a specific rule. This is emphasized by the keyword “calculate”. By evaluating fixed rules (e.g. thresholds for particular blood test results), the type of CML that is present can be derived. The UML activity for the second disease under consideration, MDS, is about twice the size of the one shown for CML. Since the basic underlying concepts are the same, we chose not to show it in this paper.

IV. PETRI NETS

There are a whole range of reasons for considering Petri nets [12] as a modeling tool for dynamic aspects of a process [13], [14]. With respect to the modeling of a diagnostic process, we opted for Petri nets (and their extensions: Coloured Petri nets (CPNs) [15]) because of their formal semantics. This, and the fact that Petri nets are well researched, results in an abundance of analysis techniques.

Additionally, they are suited for processes (e.g. CPGs) that are characterized by parallelism and synchronization [16]. The graphical nature is often brought forward as an advantage of Petri nets. In our case, the size and complexity of the resulting Petri nets are not suited for an experts’ dialog, Moreover, the modification of the model by the

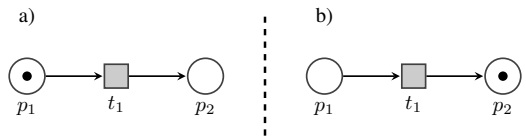

Figure 4. The net structure NST of a Petri net is given by its places p, transitions t and directed edges (arcs).

medical expert himself seems not to be feasible. As a result, we introduce translation rules, transferring a UML activity into a CPN.

A. Fundamentals

The net structure [17] of a Petri net is given by the tupel

$$NST = (P, T, F)\,, \tag{1}$$

where P is the set of places and T is the set of transitions

$$P = \{p_i : i = 1, ..., |P|\}\,,$$
$$T = \{t_i : i = 1, ..., |T|\}\,,$$

so that

$$P \cap T = \emptyset\,.$$

The flow relation F reflects the connection of places and transitions (and vice versa):

$$F \subseteq (P \times T) \cup (T \times P)\ .$$

Consequently the net structure NST of a Petri net is a directed bipartite graph (see Figure 4). To model the dynamic behavior of the system, so called "tokens" are introduced (black dots in Figure 4). The distribution of tokens on the set of places represents the state of the Petri net. It is also called marking [17]. An initial marking is given by the mapping:

$$M_0 : P \to \mathbb{N}\,.$$

Ergo, at the beginning each place p contains $M_0(p) \in \mathbb{N}$ tokens. As an example please refer to Subfigure 4a) where $M_0(p_1) = 1$ and $M_0(p_2) = 0$.

A transition that occurs (or: fires) consumes tokens respectively produces tokens in connected places. This results in new markings (see Subfigure 4b)). To allow a transition to occur, it has to be enabled. That means $M(p) \geq 1$ must hold for all places p in the pre-set ${}^\bullet\mathrm{t}$ of t using

$${}^\bullet\mathrm{t} = \{p \in P \mid (p,t) \in F\} \text{ and } t^\bullet = \{p \in P \mid (t,p) \in F\}\,,$$

where $t^\bullet$ is the post-set of t and (p,t) represents a directed edge from place p to transition t. In a step of a transition $M \xrightarrow{t} M'$ (*or:* $M[t\rangle M'$) for each place p a new marking M' is generated [17]:

$$M'(p) = \begin{cases} M(p) - 1 & \text{if } p \in {}^\bullet\mathrm{t} \text{ and } p \notin t^\bullet \\ M(p) + 1 & \text{if } p \in t^\bullet \text{ and } p \notin {}^\bullet\mathrm{t} \\ M(p) & \text{otherwise} \end{cases}\ .$$

A CPN is a so called high-level Petri net, since CPNs provide some extensions in comparison to the elementary Petri nets described above. For example the tokens can contain data. For that purpose, every token is assigned with a value (or: "colour"). Every value is of a specific type (or: "colour set"). This can be primitive types like Boolean, Integer or String – it is also possible to use composite types like Lists to store many thousand records.

Tokens having different colors are distinguishable from each other. To model this fact, a so called "multiset" (or: "bag") is needed. A multiset m over a non-empty set S is given by the mapping

$$m : S \to \mathbb{N}\,,$$

where $\mathbb{N}$ ist the set of natural numbers and $m(s)$ specifies the number of appereances of an element s in the multiset m [18]. The set of all multisets over S is denoted by S_{MS}. For example: if S is given by $S = \{\circ, \bullet\}$, the set of all multisets is $S_{MS} = \{\emptyset, \{\circ\}, \{\bullet\}, \{\circ, \bullet\}, \{\circ, \circ\}, \{\bullet, \bullet\}, \{\circ, \circ, \bullet\}, ...\}$.

Thus, a marking M of a CPN assigns every place p a multiset $m \in S_{MM}$ with $m = M(p)$ and therefore determines the state of the system. The set S defines the type that has to be specified for each place of the CPN.

In a CPN the flow of tokens can be manipulated by expressions attached to arcs and transitions. E.g., a transition can only be enabled if the guard expression evaluates to true. To specify the syntax of the CPN, it is assumed that the languagein which types, variables and expressions are stated comprises the following constructs:

- Type(v): Type of a variable v.
- Type($expr$): Type of an expression $expr$.
- Var($expr$): Set of free variables of an expression $expr$.

A CPN is then given by the following tuple [15], [18]:

$$CPN = (NST, \Sigma, V, C, G, E, I)\,, \tag{2}$$

where:

- NST is a net structure
- Σ is a finite set of types (or: non-empty colour sets).
- V is a finite set of typed variables such that Type(v)$\in \Sigma$ for all variables $v \in V$.
- $C : P \to \Sigma$ is a colour set function that assigns a type (or: Colour set) to each place.
- $G : T \to expr$ is a guard function that assigns a guard expression to each transition t such that Type($G(t)$) = Boolean and Var($expr$) $\subseteq V$.
- $E : A \to expr$ is an arc expression function that assigns an arc expression to each arc a such that Type($E(a)$) = $C(p)_{MS}$, where p is the place connected to the arc a and Var($expr$) $\subseteq V$.
- $I : P \to expr$ is an initialisation function that assigns an initialisation expression to each place p such that Type($I(p)$) = $C(p)_{MS}$ and Var($expr$) $\not\subseteq V$.

B. Translation from UML activity to Petri net

Given a UML activity represented as a graph $\mathcal{U}$ = (activitynodes, activityedges). The set of activitynodes can be further divided into different sets of nodes:

- $\mathcal{A}$: Set of actions,
- $\mathcal{S}, \mathcal{E}$: Set of initial node, set of final nodes,
- $\mathcal{B}$: Set of decision- and merge nodes (branch nodes),
- $\mathcal{C}$: Set of fork- and join nodes (concurrency nodes),
- $\mathcal{O}$: Set of object nodes.

The set of object nodes is given by the set of data pins. A node that is part of one of the node sets $\mathcal{S}, \mathcal{E}, \mathcal{B}, \mathcal{C}$ is called a control node. Furthermore, the set of activity edges is given by

- $\mathcal{KF}$: Control flow, i.e. activity edges connecting actions and control nodes, as well as edges between themselves.
- $\mathcal{DF}$: Object flow, i.e. activity edges connecting actions and object nodes or between control nodes and object nodes.

Formally the translation $[[\mathcal{U}]]$ of a UML activity $\mathcal{U}$ to a CPN is given by:

$$[[\text{(activitynodes, activityedges)}]] = (NST, \Sigma, V, C, G, E, I),$$

where NST is given by:

$$P = \mathcal{S} \cup \{\, p_{end} \,\} \cup \mathcal{B} \tag{3}$$
$$\cup \{p_e \mid e \in \mathcal{KF}, \{e.src, e.trg\} \subseteq (\mathcal{A} \cup \mathcal{C})\,\}, \tag{4}$$
$$T = \mathcal{A} \cup \mathcal{C} \tag{5}$$
$$\cup \{t_e \mid e \in \mathcal{KF}, \{e.src, e.trg\} \subseteq (\mathcal{B} \cup \mathcal{S} \cup \mathcal{E})\,\} \tag{6}$$
$$\cup \{t_e \mid e \in \mathcal{KF}, e.src \in \mathcal{S}, e.trg \in \mathcal{A}\,\} \tag{7}$$
$$\cup \{t_a \mid a \in \mathcal{A} \mid a.contains(\text{“calculate”}) = false, \exists e \in \mathcal{DF} : \{e.src, e.trg\} \cap \{a\} \neq \emptyset\}, \tag{8}$$
$$F = \{(e.src, x_e), (x_e, e.trg) \mid x_e \in P \cup T, e \in \mathcal{KF}\,\} \tag{9}$$
$$\cup \{(e.src, e.trg) \mid e \in \mathcal{KF}, e.trg \notin \mathcal{E}, e \subseteq (P \times T) \cup (T \times P)\,\} \tag{10}$$
$$\cup \{(e.src, p_{end}) \mid e \in \mathcal{KF}, e.trg \in \mathcal{E}\,\} \tag{11}$$
$$\cup \{(t_e, p_{end}) \mid e \in \mathcal{KF}, e.src \in \mathcal{S}, e.trg \in \mathcal{A}\,\} \tag{12}$$
$$\cup \{(t_a, p_{end}) \mid a \in \mathcal{A}\,\} \tag{13}$$
$$\cup \{(p_e, t_{e.target}) \mid e.target \in \mathcal{A}\,\}. \tag{14}$$

According to (3), the initial node S of the UML activity is transformed into a place of the Petri net. The activities' final nodes $\mathcal{E}$ are not translated to places – instead one single place p_{end} is added to the net structure. $\mathcal{B}$ denotes the set of branch nodes, i.e. for every decision- and merge node a place is added to the Petri net.

Furthermore (4), activity edges between two actions of $\mathcal{A}$, between two concurrency nodes of $\mathcal{C}$ or between actions and concurrency nodes are translated into places as well. Every place p_e is indexed by the corresponding activity edge e. Indexing is necessary for being able to derive the flow relation F for the newly added places. Please note that a directed edge from a source node $e.src$ to a target node $e.trg$ is given by the edge $e = (e.src, e.trg)$, where $\{e.src, e.trg\}$ denotes the set of a source- and a target node.

In (5) all actions and concurrency nodes of the UML activity are transformed into transitions of the Petri net. Additionally, in (6) transitions are added for activity edges between branch nodes, initial node and end nodes. Another transition is added for an edge from the initial node to the first action of the UML activity in (7). In our case this first action is always entitled with “suspicion of …”. This transition is needed to bypass/redirect tokens to p_{end} iff a diagnosis has been already set. In (8) further transitions are added for actions holding an input- or output pin, i.e. for actions that are the target or the source of an object flow. This procedure is only performed for actions not containing “calculate” in their name. That is because these actions represent a deterministic decision that can be implemented by the evaluation of a specific rule. All newly added transitions are indexed to allow the specification of the flow relation F. For newly added nodes $x_e \in P \cup T$ (i.e. places or transitions), the flow relation F can be derived (9) by the corresponding index e: there is an edge $(e.source, x_e)$ having node x_e as target and there is another edge $(x_e, x.target)$ having node x_e as source. The second part of the flow relation (10-11) defines edges between places and transitions that are directly translated/derived from a UML activity node. Activity edges leading to p_{end} have to be considered separately (11). That is because the resulting Petri net structure has only a single end node p_{end} – consequently all former edges leading to $\mathcal{E}$ are transformed by bending them to p_{end}. In the third part of the flow relation (12-14) the edges for the newly added transitions and places are defined. The transition from (7) and all transitions t_a are connected with p_{end} (12,13). Finally (14) adds edges between newly added places p_e and transitions.

The Σ-Algebra of the CPN (2) determines all operations, functions and types that are used in the net:

$$\Sigma = \{\ \text{STRING},$$
$$\text{PATIENT} = h(o_1) \times \cdots \times h(o_i) \times d_1 \times \cdots \times d_n$$
$$\{f_{t_e} : (\text{patient:PATIENT}) \longmapsto (\text{advice:STRING}) \mid t_e \in T,\ e \in \mathcal{KF},\ e.target = p_{end}\}\}$$

where

$$o_i \in \mathcal{O}, h(o_i) = \begin{cases} \text{INT}, & \text{if } o_i.value \in \{0, 1\} \\ \text{REAL}, & \text{if } o_i.value \in \mathbb{R}_{\geq 0} \end{cases}$$

and

$$d_n \in \{-1, 0, 1\} \text{ for } n = 1 \ldots |\mathcal{E}|.$$

In our case Σ comprises the predefined type STRING, and another type called PATIENT. This type is defined by the Cartesian product of $h(o_i)$ and the values $\{d_1, \cdots, d_n\}$.

$$G(t) = \begin{cases} f_{\text{AND}}\left((o_i \geq 0), \cdots, (o_j \geq 0)\right), & \text{if } t \in \mathcal{A} \text{ with} \\ & \{o_i, \cdots, o_j\} \in \mathcal{O}, \{(o_i, t), \cdots, (o_j, t)\} \in \mathcal{DF}, \\ f_{\text{AND}}\left((o_i \geq 0), \cdots, (o_j \geq 0)\right) = false, & \text{if } t = t_a \text{ with } a \in \mathcal{A}, \\ & \{o_i, \cdots, o_j\} \in \mathcal{O}, \{(o_i, a), \cdots, (o_j, a)\} \in \mathcal{DF}, \\ f_{\text{OR}}\left((d_i = 1), \cdots, (d_n = 1)\right), & \text{if } t = t_e : e = (e_1, e_2) \in \mathcal{KF}, e_1 \in \mathcal{S}, e_2 \in \mathcal{A}, \\ f_{\text{OR}}\left((d_i = 1), \cdots, (d_n = 1)\right) = false, & \text{if } t \in \mathcal{A} \text{ with } \exists (s, a) \in \mathcal{KF}, s \in \mathcal{S}, \\ e.guard, & \text{if } t = t_e : e = (e_1, e_2) \in \mathcal{KF}, e_1 \in \mathcal{B}, \\ & \nexists\, (v, e_1) \text{ with } v \in \mathcal{A} \cup \mathcal{B} \cup \mathcal{S}, v.contains(\text{“verify”}) \end{cases} \tag{15}$$

The function $h(o_i)$ maps object nodes o_i to the types INT or REAL. The values $\{d_1, \cdots, d_n\}$ are added for each diagnosis d_i (i.e. a UML action that is directly linked with a final node $\in \mathcal{E}$). The value -1 indicates that a diagnosis has not yet been set, whereas the values 0 and 1 are representing a negative or positive result.

The function f_{t_e} (which is assigned to a specific transition t_e) maps a variable of the type PATIENT on a variable of type STRING. If a transition t_e occurs, an advice for the medical expert of type STRING is being generated.

$$V = \{\text{ patient:PATIENT, advice:STRING }\} \tag{16}$$

$$C(p) = \begin{cases} \text{STRING}, & \text{if } p = p_{end} \\ \text{PATIENT}, & \text{otherwise} \end{cases} \tag{17}$$

V is a finite set of typed variables (2). We used a variable patient of type PATIENT and a variable advice of type STRING (16). The function $C(p)$ assigns a type (i.e. colour set) to each place p of the CPN (2). In our model, all places are of type PATIENT except for p_{end}, which is of type STRING (17).

The guard function $G(t)$ assigns a guard to each transition $t \in T$. The guard is an expression which can evaluate to $true$ or $false$. In the latter case the transition can not be enabled and therefore is not able to occur.

If a transition has been directly generated from a UML action (i.e. $t \in \mathcal{A}$), the guard function assigns the guard expression $f_{\text{AND}}\left((o_i \geq 0), \cdots, (o_j \geq 0)\right)$ (15). Where f_{AND} uses the Boolean expression AND to combine the arguments of the function: $(o_i \geq 0), \cdots, (o_j \geq 0)$. The set $\{o_i, \ldots, o_j\}$ represents those object nodes, that are connected to the corresponding action. That is, there is a directed edge from object node o_i to action t: (o_i, t). Or in other words: only if all values of an examination have been set, the corresponding transition can be enabled and therefore can occur.

$$E(f) = \begin{cases} \text{1'advice}, & \text{if } f.target = p_{end} \\ \text{1'patient}, & \text{otherwise} \end{cases} \tag{18}$$

$$I(p) = \begin{cases} \text{1'patient}, & \text{if } p \in \mathcal{S} \\ \text{emptymultiset}, & \text{otherwise} \end{cases} \tag{19}$$

$E(f)$ assigns an expression to all edges $f \in F$ (18). If the target of the edge is p_{end} (which is of type STRING), the resulting type of the assigned expression must evaluate to STRING, too. In all other cases, a patient is bound to a value of type PATIENT. $I(p)$ assigns an initialization expression to all places $p \in P$ and therefore specifies the initial marking of the CPN (19). The place p that corresponds to the UML initial node contains one token, whereas all other places are empty (i.e. they are assigned to an empty multiset).

V. Verification and validation

To validate the Petri net model based on the CPG of CML, we used a patient record from our former work [2]. Some examination values were removed, namely “neutrophilic leukocytes” (originated from differential blood count), “philadelphia chromosome” (originated from cyto genetics) and bcr-abl (originated from molecular genetics).

The token containing the modified patient record is put on the start place of the CPN. Because the diagnosis is not set and all examination values of “anamnesis” (case history), “physical examination” and “complete blood count (CBC) without differential” are present within the token – the corresponding transitions can occur. The token arrives at the pre-set of the transitions “differential blood count” and “suggest differential blood count” (see Subfigure 5a)). Because the examination value “neutrophilic leukocytes” is missing, the guard expression for “differential blood count” evaluates to $false$ and therefore this transition is not enabled. The transition “suggest differential blood count” is enabled since the guard expression evaluates to $true$. By firing this transition a new token of type STRING is generated at the final place of the CPN (see Subfigure 5b)). The token contains the recommendation which examination value to take next to complete the patient details according to the guideline model.

If a differential blood count has been performed, the patient details and the corresponding token are updated. To generate a new recommendation, the token is put on the start place again. Because the value of “neutrophilic leukocytes” is set, the corresponding transitions can occur.

Now, tokens appear in the pre-set of the transitions “bone marrow (BM) cytology”, “suggest molecular genetics” and

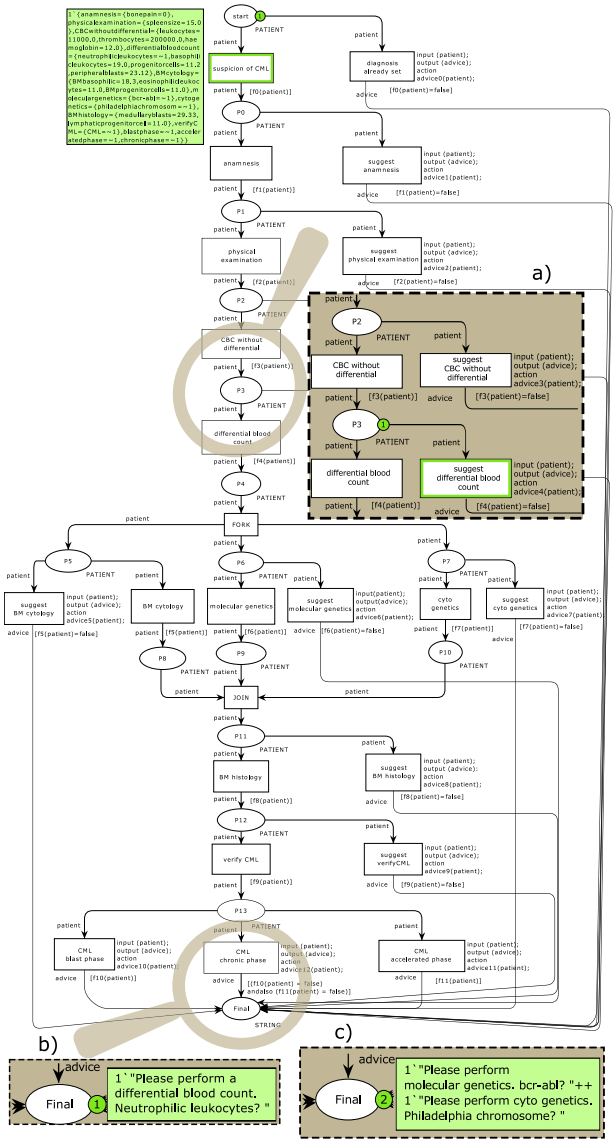

Figure 5. Resulting CPN after applying translations rules to the UML activity in Figure 3. Subfigures (a-c) depict the flow of tokens for a patient with fragmentary examination values.

"suggest cyto genetics", all of which can occur. By firing, two advices are generated, i.e. two tokens of type STRING are created in the final place of the CPN (see Subfigure 5c)).

VI. Conclusion

In this paper we introduced a framework to facilitate the implementation of diagnostic processes (the processes are given by CPGs). UML activities are used as an interface for the models that provide the actual assistance functions.

The assistance function provided by a Petri net guides the medical practitioner through the whole diagnostic process. In case of fragmentary patient details, it is able to recommend suitable examination values needed to complete the assessment suggested by the CPG. In the near future, our approach will be integrated in a real world diagnostic application that helps to overcome barriers in CPG implementation and reduces the gap between theoretical medical knowledge and practical solutions.

References

[1] National Center for Biotechnology Information, U.S. National Library of Medicine, "Pubmed," September 2015, Accessed: 2015/09/16. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed

[2] P. Philipp, Y. Fischer, D. Hempel, and J. Beyerer, "Modeling of clinical practice guidelines for interactive assistance in diagnostic processes," in *WorldComp 2015, World Congress in Computer Science, Computer Engineering, and Applied Computing : HIMS 2015, International Conference on Health Informatics and Medical Systems, july 27-30, Las Vegas, Nevada, USA*, 2015, pp. 3–9.

[3] E. Steinberg, S. Greenfield, M. Mancher, *et al.*, *Clinical Practice Guidelines We Can Trust*. National Academies Press, 2011.

[4] E. Field and K. Lohr, *Guidelines for Clinical Practice: From Development to Use*. National Academies Press, 1992.

[5] P. Philipp, Y. Fischer, D. Hempel, and J. Beyerer, "Framework for an interactive assistance in diagnostic processes based on probabilistic modeling of clinical practice guidelines," in *Emerging Trends in Computational Biology, Bioinformatics, and Systems Biology (In Press)*. Elsevier, 2016.

[6] H. Kirchner, M. Fiene, and G. Ollenschläger, "Bewertung und Implementierung von Leitlinien," *Die Rehabilitation*, vol. 42, no. 2, pp. 74–82, 2003.

[7] H. Störrle, "Semantics of control-flow in UML 2.0 activities," in *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on*, 2004, pp. 235–242.

[8] H. Störrle and J. Hausmann, "semantics of uml 2.0 activities," in *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, 2004.

[9] H. Störrle, "Semantics and verification of data flow in UML 2.0 activities," *Electronic Notes in Theoretical Computer Science*, vol. 127, no. 4, pp. 35–52, 2005.

[10] OMG, "OMG Unified Modeling Language(OMG UML) Superstructure Version 2.4.1," 2011, Accessed: 2014/08/28. [Online]. Available: http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/

[11] C. Rupp, S. Queins, *et al.*, *UML 2 glasklar*. Mnchen: Carl Hanser Verlag GmbH Co KG, 2012.

[12] C. A. Petri, "Kommunikation mit Automaten," Ph.D. dissertation, Universität Bonn, 1962.

[13] W. van der Aalst, "Three good reasons for using a Petri-net-based workflow management system," in *Proceedings of the International Working Conference on Information and Process Integration in Enterprises*, 1996, pp. 179–201.

[14] W. van der Aalst, "The application of Petri nets to workflow management," *Journal of circuits, systems, and computers*, vol. 8, no. 01, pp. 21–66, 1998.

[15] K. Jensen, "Coloured Petri nets and the invariant-method," *Theoretical computer science*, vol. 14, pp. 317–336, 1981.

[16] F. DiCesare, G. Harhalakis, J.-M. Proth, M. Silva, and F. Vernadat, *Practice of Petri nets in manufacturing*. Springer, 1993.

[17] W. Reisig, *Understanding Petri Nets*. Springer, 2013.

[18] K. Jensen and L. M. Kristensen, *Coloured Petri Nets*. Dordrecht: Springer, 2009.