

# Using the Petri Net Markup Language for Exchanging Business Processes – Potential and Limitations –

Ekkart Kindler

Software Engineering Group  
Computer Science Department  
University of Paderborn  
D-33095 Paderborn  
Germany  
kindler@upb.de

**Abstract** The *Petri Net Markup Language* (PNML) is an XML-based interchange format for Petri nets. Its focus is on universality and flexibility, which is achieved by a technique for defining new Petri net types through *Petri Net Type Definitions* (PNTDs).

Many business process modelling techniques are based on Petri nets. Since PNML provides a means for defining Petri net types, it might be a worthwhile project to define a PNTD for business process models, which supports the exchange of business process models among different BPM tools.

In this paper, we discuss the potential and the limitations of PNML for exchanging business processes. Moreover, we discuss some lessons learned from the standardization of PNML, which could be helpful for the standardization of interchange formats for business process models in general.

## 1 Introduction

The *Petri Net Markup Language* (PNML) is a widely accepted XML-based interchange format for Petri nets [JKW00a, WK03b, BCvH<sup>+</sup>03], which is currently standardized as ISO/IEC 15909-2. The main problem with devising a standard interchange format for Petri nets is the multitude of existing versions and variants of Petri nets – almost every tool uses a slightly different version of Petri nets, and new versions and variants of Petri nets are invented every year. In order to tackle this problem, PNML provides interfaces for defining new features and new types of Petri nets: the *Features Definition Interface* and the *Type Definition Interface*.

There is an even greater variety of formalisms and notations for business process modelling. Therefore, devising an interchange format for business process models is a much more challenging task. This task, however, is beyond the scope of this paper. But, there are many notations for business models that are based on Petri nets or have an underlying Petri net semantics. Thus, it might be a worthwhile task to devise a Petri Net Type Definition (PNTD) for PNML that supports the exchange of Petri net based business process models.

In this paper, we outline how such a PNTD for PNML could look like<sup>1</sup> and how the corresponding interchange format could be used, and we discuss the potential and the limitations of such an approach. In addition, we will summarize our experiences and the lessons learned during the standardization of PNML, which could be helpful for the standardization of exchange formats for business process models.

The paper is structured as follows: We first give an overview on the principles and the concepts of PNML and, in particular, present its meta model, its XML-syntax, and its Type Definition Interface. Then, we will discuss how PNML could be used for exchanging business processes. In the end, we discuss the lessons learned during the (ongoing) standardization of PNML.

## 2 PNML

As mentioned in the introduction, PNML was designed to support all kinds, versions, and variants of Petri nets. In order to achieve this flexibility, PNML is split into several parts and is equipped with interfaces for defining new features and new types of Petri nets.

### 2.1 Overview

The different parts of PNML and their relationships are shown in Fig. 1. The *meta model* defines the basic structure of a PNML file; the *Type Definition Interface* allows the definition of new Petri net types that restrict the legal files of the meta model; and the *Feature Definition Interface* allows the definition of new features for Petri nets. These three parts are fixed once and for all. Another part of PNML, the *Conventions Document*, evolves. It contains the definition of a set of standard features of Petri nets, which are defined according to the feature definition interface. Moreover, there will be several *Standard Petri Net Types*, using some features from the Conventions Document and possibly others. New features and new types can be added to the Conventions Document and to the standard types when they are of common interest. Due to their evolving nature, these documents are best published

---

<sup>1</sup>We will not present a concrete definition of this format here in order to avoid the publication of a premature definition, which could easily result in a failure of the complete idea (see Sect. 4.3 for details).

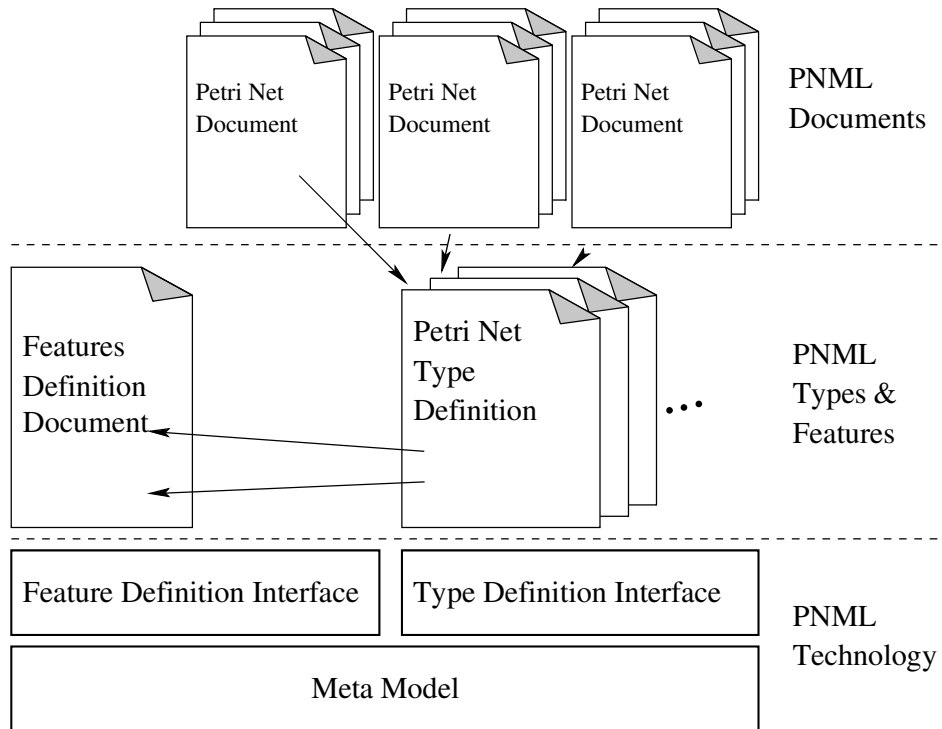


Figure 1: Overview of the parts of PNML

and maintained via a web site. Up to now, only the technological aspects of these documents are defined. The precise process of maintaining these documents, and when and how to update them is not yet fixed. But, it will be in close coordination with the Steering Committee of the annual International Conference on the Application and Theory of Petri nets, which is the major scientific event in the field of Petri nets.

## 2.2 Meta model

Figure 2 shows that part of the meta model of PNML in UML notation, which is relevant in our context. For the full meta model, we refer to [BCvH<sup>+</sup>03]. We will explain the meta model below.

### 2.2.1 Petri nets and objects

A document that meets the requirements of PNML is called a *Petri net document*; it may contain several *Petri nets*. Each Petri net consists of *objects*, which, basically, represent the graph structure of the Petri net. Each object within a Petri net document has a unique *identifier*, which can be used to refer to this object. In

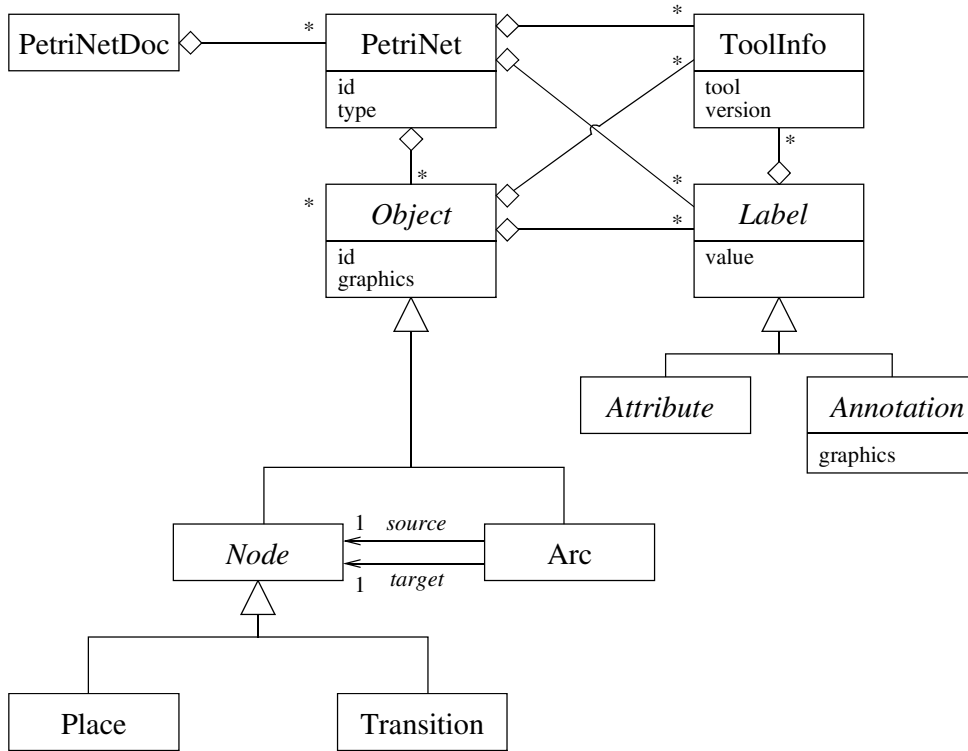


Figure 2: The PNML meta model

*basic PNML*<sup>2</sup> an object is a *place*, a *transition* or an *arc*. For convenience, a place or a transition is called a *node*.

### 2.2.2 Labels

In order to assign further meaning to an object, each object may have *labels*. Typically, a label represents the name of a node, the initial marking of a place, the guard of a transition, or the inscription of an arc. In addition, the Petri net itself may have some labels. For example, the declarations of functions and variables that are used in the arc inscriptions could be labels of a high-level Petri net. The legal labels and the legal combinations of labels are defined by the Petri net type. The type of a Petri net is defined by a reference to a unique *Petri Net Type Definition* (PNTD), which will be discussed in Sect. 2.4.

It turned out that two kinds of labels should be distinguished: *annotations* and *attributes*. But this distinction is not relevant for this paper, so we do not discuss this issue here (see [WK03b, BCvH<sup>+</sup>03] for details).

<sup>2</sup>Basic PNML refers to the version of PNML without any structuring mechanism, which is the version discussed here. In more evolved versions, a net may consist of pages and modules, too.

### 2.2.3 Graphical information

Each object and each annotation is equipped with graphical information. For a node, this information includes its position; for an arc, it includes a list of positions that define intermediate points of the arc; for an annotation, it includes its relative position with respect to the corresponding object. Additionally, there can be information concerning the size, colour, and shape of a node or an arc, or concerning the colour, font, and font size of a label. Another graphical information can be an image for displaying a node.

### 2.2.4 Tool specific information

For some tools, it might be necessary to store tool specific information, which is not supposed to be used by other tools. In order to store this information, each object and each label may be equipped with *tool specific information*. Its format depends on the tool and is not specified by PNML. PNML provides a mechanism for clearly marking tool specific information along with the name and the version of the tool that added this information. Therefore, other tools can easily ignore it, and adding tool specific information will never compromise a PNML file.

### 2.2.5 Pages and modules

More advanced versions of PNML provide mechanisms for structuring Petri nets. In *structured PNML*, a single Petri net can be drawn on several *pages*. In *modular PNML*, there is a concept for defining and instantiating modules, which can be used for constructing systems in a modular and hierarchical way. For more details on structured and modular PNML, we refer to [WK03b, BCvH<sup>+</sup>03]. The interesting aspect of the page and module concept is that they are completely independent of the Petri net type. They can be used with any type. This way, PNML implements a module concept for any version of Petri nets – even when these versions do not have these concept on their own.

## 2.3 XML representation

The PNML meta model is translated into XML syntax in a straightforward manner. Technically, the syntax of PNML is defined by a RELAX NG grammar [Relax], which can be found on the PNML web site [PNML].

Class	XML element	XML Attributes
PetriNetDoc	<code>&lt;pnml&gt;</code>	
PetriNet	<code>&lt;net&gt;</code>	id: ID type: anyURI
Place	<code>&lt;place&gt;</code>	id: ID
Transition	<code>&lt;transition&gt;</code>	id: ID
Arc	<code>&lt;arc&gt;</code>	id: ID source: IDRef (Node) target: IDRef (Node)
ToolInfo	<code>&lt;toolspecific&gt;</code>	tool: string version: string
Graphics	<code>&lt;graphics&gt;</code>	

Table 1: Translation of the PNML meta model into PNML elements

### 2.3.1 PNML elements

Here, we present the XML syntax in a more compact way: Basically, each concrete class<sup>3</sup> of the PNML meta model is translated into an XML element. This translation along with the attributes and their data types is given in Tab. 1. These XML elements are the *keywords* of PNML and are called *PNML elements* for short. For each PNML element, the aggregations of the meta model define in which elements it may occur as a child element. Note that we have omitted the class Graphics from the meta model in Fig. 2 so as not to clutter the diagram. The classes with associated graphical information are instead indicated by an attribute “graphics”.

The data type ID in Tab. 1 defines the declaration of an identifier, which must be unique within the PNML file. The data type IDRef defines references to these identifiers.

### 2.3.2 Labels

There are no PNML elements for labels because the meta model does not define any concrete label. Concrete labels are defined by the Petri net types. An XML element that is not defined in the meta model (i. e. not occurring in Tab. 1) is considered as a label of the PNML element in which it occurs. For example, an `<initialMarking>` element could be a label for a place, which represents its initial marking. Likewise `<name>` could represent the name of an object, and `<inscription>` could represent an arc inscription. A legal element for a label may consist of further elements. The value of a label appears as a string in a `<text>` element. Furthermore, the value may be represented as an XML tree in a `<structure>` element. An optional PNML `<graphics>` element defines its graphical appearance, and further optional PNML `<toolspecific>` elements may add tool specific information to the label.

<sup>3</sup>A class in a UML diagram is concrete if its name is not displayed in italics.

Parent element class	Sub-elements of <code>&lt;graphics&gt;</code>
<i>Node</i>	<code>&lt;position&gt;</code> (required) <code>&lt;dimension&gt;</code> <code>&lt;fill&gt;</code> <code>&lt;line&gt;</code>
<i>Arc</i>	<code>&lt;position&gt;</code> (zero or more) <code>&lt;line&gt;</code>
<i>Annotation</i>	<code>&lt;offset&gt;</code> (required) <code>&lt;fill&gt;</code> <code>&lt;line&gt;</code> <code>&lt;font&gt;</code>

Table 2: Elements in the `<graphics>` element depending of the parent element

### 2.3.3 Graphics

PNML elements and labels include graphical information. The structure of the PNML `<graphics>` element depends on the element in which it appears. Table 2 shows the elements which may occur in the substructure of a `<graphics>` element. The `<position>` element defines an absolute position and is required for each node, whereas the `<offset>` element defines a relative position and is required for each annotation. The other sub-elements of `<graphics>` are optional. For an arc, the (possibly empty) sequence of `<position>` elements defines its intermediate points. Each absolute or relative position refers to Cartesian coordinates  $(x, y)$ . As for most graphical tools, the  $x$ -axis runs from left to right and the  $y$ -axis from top to bottom. More details on the effect of the graphical features can be found in [BCvH<sup>+</sup>03].

### 2.3.4 Example

In order to illustrate the structure of a PNML file, we consider the simple example net shown in Fig. 3. Listing 1 shows the corresponding PNML code.

It is a straightforward translation, where we have labels for the names of objects, for the initial markings, and for arc inscriptions. Note that we assume that the dashed outline of the transition results from the tool specific information `<hidden>` from an imaginary tool *PN4all*. This appearance is not defined in PNML; it comes from the particular (imaginary) tool.

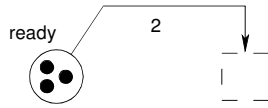


Figure 3: A simple P/T-system

Listing 1: PNML code of the example net in Fig. 3

```

<pnml xmlns="http://www.example.org/pnml">
  <net id="n1" type="http://www.example.org/pnml/PTNet">
    <name>
      <text>An example P/T-net</text>
5    </name>
    <place id="p1">
      <graphics>
        <position x="20" y="20"/>
      </graphics>
10    <name>
      <text>ready</text>
      <graphics>
        <offset x="-10" y="-8"/>
      </graphics>
15    </name>
      <initialMarking>
        <text>3</text>
      </initialMarking>
    </place>
20    <transition id="t1">
      <graphics>
        <position x="60" y="20"/>
      </graphics>
      <toolspecific tool="PN4all" version="0.1">
25        <hidden/>
      </toolspecific>
    </transition>
    <arc id="a1" source="p1" target="t1">
      <graphics>
30        <position x="30" y="5"/>
        <position x="60" y="5"/>
      </graphics>
      <inscription>
        <text>2</text>
35        <graphics>
          <offset x="15" y="-2"/>
        </graphics>
      </inscription>
    </arc>
40  </net>
</pnml>

```

## 2.4 Petri Net Type Definition

Next, we discuss the definition of a Petri net type. In order to define a type, we need to define labels first.

### 2.4.1 Label definition

Listing 2 shows the RELAX NG definition of the label `<initialMarking>`, which represents the initial marking of a place of a P/T-system (cf. List. 1). Its value (in a `<text>` element) should be a natural number, which is formalized by referring to the corresponding data type `nonNegativeInteger` of the data type system of XML Schema [XSch]. Note that the optional graphical and tool specific information do not occur in this label definition; this is not necessary, because these standard elements for annotations in the meta model of PNML are included from the definition of the standard annotation content. Such label definitions can either be given explicitly for each Petri net type, or they can be included in the Conventions Document, such that Petri net type definitions can refer to these definitions.

Listing 2: Label definition

```
<define name="PTMarking"
  xmlns:pnml="http://www.informatik.hu-berlin.de/top/pnml">
  <element name="initialMarking">
    <interleave>
5      <element name="text">
          <data type="nonNegativeInteger"
            datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>
        </element>
        <ref name="pnml:StandardAnnotationContent"/>
10    </interleave>
    </element>
  </define>
```

### 2.4.2 Petri Net Type Definitions (PNTDs)

Listing 3 shows the Petri Net Type Definition (PNTD) for P/T-Systems as a RELAX NG grammar. Firstly, it includes both the definitions from the meta model of PNML (`pnml.rng`) and the definitions from the Conventions Document (`conv.rng`), which, in particular, contains the definition from List. 2, a similar definition for arc inscriptions of P/T-systems, and a definition for names.

Secondly, the PNTD defines the legal labels for the whole net and the different objects of the net. In our example, the net and the places may have an annotation for names. Furthermore, the places are equipped with an initial marking and the

Listing 3: PNTD for P/T-Systems

```

<grammar ns="http://www.example.org/pnml"
  xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:conv="http://www.informatik.hu-berlin.de/top/pnml/conv">
  <include href="http://www.informatik.hu-berlin.de/top/pnml/pnml.rng"/>
5  <include href="http://www.informatik.hu-berlin.de/top/pnml/conv.rng"/>
  <define name="NetType" combine="replace">
    <text>http://www.example.org/pnml/PTNet</text>
  </define>
  <define name="Net" combine="interleave">
10  <optional><ref name="conv:Name"/></optional>
  </define>
  <define name="Place" combine="interleave">
    <interleave>
      <optional><ref name="conv:PTMarking"/></optional>
15  <optional><ref name="conv:Name"/></optional>
    </interleave>
  </define>
  <define name="Arc" combine="interleave">
    <optional><ref name="conv:PTArcInscription"/></optional>
20  </define>
</grammar>

```

arcs are equipped with an inscription. Note that all labels are optional here. The labels are associated with the net objects by giving a reference to the corresponding definitions in the Conventions Document. Technically, the definition extends the original definition of the net, places and arcs of the RELAX NG grammar for PNML.

## 2.5 Related work

Of course, there are XML-based formats and other technologies that could be used for interchanging Petri nets. For example, *XMI* [OMG03] could be used for ‘mapping’ PNML’s meta model to XML, or *GXL* [HWS00] could be used for exchanging Petri nets as a graphs.

Besides the fact that all these formats and technologies were developed concurrently, there are some reasons for having a dedicated format for Petri nets. One reason for not using GXL is that PNML can exploit some features that are specific to Petri nets. For example, this applies to the module concept of PNML, which is very specific to Petri nets. One reason for not using XMI is that PNML should be easily usable without being familiar to and without necessarily using XMI technology.

A more detailed discussion of the principles governing the design of PNML can be found in [JKW00b, BCvH<sup>+</sup>03, WK03b].

### 3 Business process modelling

In the previous section, we have introduced the basic principles and concepts of PNML. Next, we will discuss in which way PNML could be used as an interchange format for business processes. To this end, we briefly rephrase our understanding of *business processes* and *business process models*, which resembles that of the Workflow Management Coalition [Hol95, WFM99].

#### 3.1 Business processes and their models

A *business process* consist of a set of activities (or tasks) that are executed in some enterprise or administration according to some rules in order to achieve certain goals. The execution of an activity may require an agent or some resources and some information.

A *business process model* more or less exactly captures the rules according to which a specific class of business processes is executed and, in particular, defines which activities are to be performed, in which order they are to be performed, by which agents and resources they are to be performed, and in which way information or documents are used and propagated among activities.

It turned out that business processes have different aspects that can be modelled independently of each other. The *behavioural aspect* models the order in which the different activities must be executed, the *organizational aspect* models the agents and resources of an enterprise and how they are associated with certain activities, and the *informational aspect* models the information used in the process, how the information is represented, and how the information is propagated among the activities.

Actually, there are even more aspects, in particular, when it comes to the automatic execution of business processes by a *workflow management system*. Also *timing aspects* may be very important for doing performance analysis and business process re-engineering. But, we do not go into the details of these aspects here.

By using the term *business process modelling*, people refer to quite different things. Some refer to the modelling of the behavioural aspect only (which is typical for people from the Petri net community), whereas others refer to all aspects mentioned above; some people even refer to the models executed by a workflow management system – in that case they are usually called *workflow models* [LR99, Hol95].

Likewise, the *degree of rigor* of business process models varies. For some people, they are just informal and even incomplete sketches that help giving a rough understanding of an enterprise's or administration's business processes. For others, they are completely formal such that precise analysis or even execution is possible.

There is no harm in the different notions or perceptions of business process modelling. All of them have their benefits and their uses with respect to certain ob-

jectives. But, when it comes to interchange formats for business process models, we need to be aware of the different perceptions of business process modelling and the different objectives. To us it is not clear at all, whether there can be or there should be a single interchange format serving all purposes and perceptions in business process modelling.

### 3.2 Petri nets

Petri nets are well-known for being a formal and rigorous modelling technique. Therefore, Petri nets are well-suited for rigorous business process modelling. Classical Petri nets such as Place/Transition-Systems (P/T-Systems) cannot distinguish between different kinds of token (there are black tokens only) and there is no means for modelling data. This is the reason why Petri nets are usually used for modelling the behavioural aspect of business processes only. Maybe, the best known approach is the one by van der Aalst, in which he proposes a special kind of unmarked P/T-Systems with a single input place and a single output place, called *workflow nets* [vdA97, vdAvH02]. Sometimes, however, Petri nets are considered to lack modelling power. Van der Aalst himself proposes some extensions called *YAWL* [vdAtH02].

Classical Petri nets are also good for analysis and planning of resource assignment to tasks and for business processes in particular. When using timed or stochastic versions, Petri nets can also be used for doing performance analysis: And these techniques turn out to be quite effective. But, this does not necessarily mean that Petri nets are good for modelling the organizational aspects of business processes. In fact, they are not good at all<sup>4</sup>. When it comes to realistic processes and organizational models, the corresponding Petri net become quite complex and inscrutable. So it would be much better to use a more appropriate notation for modelling the organization and the relation of its resources to the activities. Then we could automatically translate this model to a Petri net, which can be used for automatic analysis. In order to make this approach work, an interchange format for the organization aspect of business processes must have a clean and simple meta model and a mechanism for associating the resources of the organizational model with the activities in the behavioural model. This notation could and should be independent of the particular modelling formalism for the behavioural aspect.

In classical Petri nets, the information aspect of a business process cannot be modelled. But, there are different versions of high-level Petri nets that allow us to model information and data. Actually, the information model could be defined in almost any notation; then, a high-level Petri net could be used for modelling the propagation of the information, resp. data or documents between different activities. Again, (high-level) Petri nets may not be good at modelling the structure

---

<sup>4</sup>Too often, the distinction between being a good modelling notation and providing a good analysis techniques is not clearly made. This might be the source of many misunderstandings in discussions on appropriate and inappropriate modelling and analysis techniques.

of the data or in defining the information model – most of them use a notation that is not specific to Petri nets at all. But, Petri nets are good at modelling how information is propagated between different activities, once the information model is defined.

### **3.3 PNML for business process models**

As indicated in the previous section already, there are several ways in which PNML could be used as an interchange format for business process models.

#### **3.3.1 Workflow nets**

First, PNML could be used as an interchange format for the behavioural aspect of business processes only. To this end, we could define a PNTD for *workflow nets* as defined by van der Aalst [vdAvH02]. This PNTD would essentially be the PNTD for P/T-Systems without the labels representing the initial marking; additionally, there would be a distinguished start and end place.

The corresponding PNTD is very simple. But, we do not give a PNTD for workflow nets, here. With this PNTD, PNML could be used for exchanging workflow nets.

#### **3.3.2 Resources**

Of course, we can easily extend the PNTD for workflow nets with some labels that define the needed resources for each activity (resp. the transition representing it). The syntax of these labels, however, will depend on the modelling notation for the organizational model resp. its underlying meta model. Once the meta model resp. the interface for referring to an organization model is fixed, it should be an easy task to define the syntax of the resource labels.

#### **3.3.3 Information propagation**

Similarly, we could easily devise a PNTD for high-level Petri nets in order to model the propagation of information between different activities. The concrete syntax of the arc-labels would depend on the meta model resp. its interface of the information aspect.

#### **3.3.4 Extensions**

For particular purposes, we could easily extend the format with all necessary features that are available in the Conventions Document of PNML. For example, we could use labels for timed Petri nets or for stochastic Petri nets in order to represent timing information.

### 3.3.5 Interfaces

In essence, a PNTD for defining business processes would be a definition of workflow nets, equipped with some additional information. For defining the concrete labels for the informational aspect and for the organizational aspect, however, it would be necessary to define interfaces resp. meta models for these aspects first. These interfaces, however, could and should be completely independent of Petri nets or any other model for the behavioural aspect. Therefore, discussing and fixing models for these aspects is more important than devising a PNTD for workflow nets. Once the meta models resp. interfaces for the other aspects are defined, it is straightforward to define a PNTD for the behavioural aspect of business process models.

## 4 Lessons learned with PNML

The discussion of XML-based interchange formats for Petri nets started about four years ago. And the standardization process is not finished yet; in fact, the official standardization as ISO/IEC 15909-2 has just begun. So, we are still learning some lessons on the standardization of PNML.

Nevertheless, we report on some of the lessons learned to date in the hope that they might be helpful in the standardization of an interchange format for business processes. But, we are aware that the standardization of business processes is a much more difficult issue.

### 4.1 Organization

The standardization of an XML-based<sup>5</sup> interchange format for Petri nets was started in 2000 with a workshop at the annual International Conference on Application and Theory of Petri Nets [BBK<sup>+</sup>00]. This and all subsequent activities and events were closely coordinated with the Steering Committee of the International Conference on Application and Theory of Petri Nets. Though there have been concrete proposals for exchange formats for Petri nets at the first workshop, the focus of this workshop and the discussions were on the principles and objectives of such an interchange format and on ideas how to deal with the variety of different Petri net types. At this workshop, we agreed that a standard interchange format would be helpful even if not all information can be interchanged among different tools. A format that would help to interchange the basic structure of a net among different tools would be helpful in many cases.

---

<sup>5</sup>Actually, there have been other interchange formats for particular versions of Petri nets long before that. For example, there was the *Abstract Petri Net Notation (APNN)* [BKK95].

After the first workshop in 2000, we had more or less informal meetings every year during the annual International Conference on Application and Theory of Petri Nets, which were open to everybody. Finally, these meetings resulted in a joint paper on PNML [BCvH<sup>+</sup>03], which will be the basis for the standard ISO/IEC 15909-2. The lesson learned from this is that it is important to have a common understanding of the objectives and the purposes of an interchange format before making proposals for such a format.

## 4.2 Principles

During the first workshop, it became clear that dealing with the variety of different versions and dialects of Petri nets would be one of the main issues in a new standard. Therefore, openness and extensibility of the format were the main principles driving the design of PNML.

PNML provides extensibility in several ways. One way is the features and type concepts, which allows us to define new labels for Petri nets and new Petri net types. It will be an ongoing activity to update and to maintain the standard features and the standard Petri net types. But, everybody is free to define an own Petri net type – other tools will be able to understand at least that part that refers to standard features.

Another way for providing extensibility is tool specific information. Every tool is allowed to store its private information for the different objects within a Petri net. This way, all information that is not covered by PNML can be stored in a PNML file, and tools are not forced to have their private format in addition to PNML. A tool can store any information necessary within a PNML file. On the one hand, this is an important feature for tool providers to support PNML. On the other hand, it is also a dangerous feature: If different tools store important information as tool specific information only, this information cannot be used by other tools anymore. Therefore, we discourage the use of tool specific information wherever possible and we try to make sure that there are standard features and standard Petri net types for all important features and Petri net types available.

A similar problem might occur concerning graphics. Initially, we had only very few graphical features in PNML and considered all other graphical information to be tool specific. But, we realized that this way most graphical information might go to tool specific information and, therefore, would be lost for other tools. Therefore, we now aim at providing a way to represent all graphical features that are well-established in Petri net tools within PNML itself<sup>6</sup>. Only very special or fancy graphical features cannot be expressed in PNML. This way, PNML not only guarantees the same appearance of a Petri net in different tools, we can even provide

---

<sup>6</sup>In fact, many people are still not aware of the much richer graphical features of the current version of PNML because they had a look to now outdated versions only. This is another argument for not starting with a half-baked definition.

a standard transformation of PNML files to SVG [FJe03]. Actually, we define the graphical appearance of a PNML file by an XSLT transformation to SVG, which was first proposed in [Ste02].

### 4.3 Tools

Fortunately, many tool providers started to implement PNML, once the first draft of PNML was published. Maybe, this was the most important factor to the success of PNML. But, one problem was that the first papers focused on the concepts of PNML and were incomplete in some technical issues<sup>7</sup>. Therefore, there have been different variants and versions of PNML before we came up with a complete version. Moreover, we had to change the definition of some constructs in order to get a clearer design of PNML. Though these could be considered to be minor changes only, the tools using PNML already ran into problem.

The lesson learned from this is that already the first draft should be worked out in full technical detail and a validation tool should be available right from the beginning. This can help avoiding frustration of tool providers, who are so important for the success of a standard. This is the reason why we do not provide a concrete definition of a PNTD for business process models here.

## 5 Conclusion

In this paper, we have given an overview on PNML, and we discussed how PNML could be used for exchanging business process models.

The idea boils down to providing a Petri Net Type Definition (PNTD) for the behavioural aspect of business process models. Other aspects could be incorporated into this model too. But, we do not propose to use Petri nets for modelling the informational or the organizational aspect of a business process. Rather, we would like to use some more appropriate formalism, which provides a clear interface such that Petri net models can refer to these models in order to associate resources with activities and to model the propagation of data based on these models.

In a nutshell, we propose to start with a standardization of the interfaces of the meta models for the different aspects of business process models, rather than to start with a format for business processes themselves. Once, these interfaces are defined, devising a PNTD for the behavioural aspect of business processes will be easy.

---

<sup>7</sup>We must admit that our own tool, the *Petri Net Kernel* [WK03a], implemented PNML slightly different from its specification.

## Acknowledgments

Some parts of this papers are taken from the PNML paper [BCvH<sup>+</sup>03] with only minor modifications. I would like to thank all co-authors of that paper for their permission to use these parts and for their encouragement to carry on. In particular, I would like to thank Michael Weber and Renier Post for their comments and suggestions on an earlier version of this paper. Moreover, I would like to thank the anonymous reviewers for their comments and questions, which helped to improve the presentation of this material.

## References

- [BBK<sup>+</sup>00] R. Bastide, J. Billington, E. Kindler, F. Kordon, and K. H. Mortensen, editors. *Meeting on XML/SGML based Interchange Formats for Petri Nets*. University of Aarhus, Department of Computer Science, June 2000.
- [BCvH<sup>+</sup>03] Jonathan Billington, Søren Christensen, Kees van Hee, Ekkart Kindler, Olaf Kummer, Laure Petrucci, Reinier Post, Christian Stehno, and Michael Weber. The Petri Net Markup Language: Concepts, Technology, and Tools. In W. van der Aalst and E. Best, editors, *Application and Theory of Petri Nets 2003, 24<sup>th</sup> International Conference, LNCS 2679*, pages 483–505. Springer, June 2003.
- [BKK95] Falko Bause, Peter Kemper, and Pieter Kritzinger. Abstract Petri Net Notation. *Petri Net Newsletter*, 49:9–27, October 1995.
- [FJe03] J. Ferraiolo, F. Jun, and D. Jackson (eds.). Scalable Vector Graphics (SVG) 1.1 Specification. URL <http://www.w3.org/TR/SVG11/>, 2003.
- [Hol95] David Hollingsworth. The Workflow Reference Model. Technical Report TC00-1003, The Workflow Management Coalition (WfMC), January 1995.
- [HWS00] Richard C. Holt, Andreas Winter, and Andy Schürr. GXL: Towards a Standard Exchange Format. In *7<sup>th</sup> Working Conference on Reverse Engineering*, pages 162–171. IEEE Computer Society, 2000.
- [JKW00a] Matthias Jünger, Ekkart Kindler, and Michael Weber. Towards a Generic Interchange Format for Petri Nets – Position Paper. In R. Bastide, J. Billington, E. Kindler, F. Kordon, and K. H. Mortensen, editors, *Meeting on XML/SGML based Interchange Formats for Petri Nets*, pages 1–5, June 2000.
- [JKW00b] Matthias Jünger, Ekkart Kindler, and Michael Weber. The Petri Net Markup Language. *Petri Net Newsletter*, 59:24–29, October 2000.
- [LR99] Frank Leymann and Dieter Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.
- [OMG03] XML Metadata Interchange (XMI) Specification, Version 2.0. Technical Report formal/03-05-02, The Object Management Group, Inc., May 2003.
- [PNML] The Petri Net Markup Language. URL <http://www.informatik.hu-berlin.de/top/pnml/>. 2001/07/19.
- [Relax] RELAX NG Specification. URL <http://www.oasis-open.org/committees/relax-ng/>. 2001/12/03.
- [Ste02] Christian Stehno. Petri Net Markup Language: Implementation and Application. In J. Desel and M. Weske, editors, *Promise 2002, Lecture Notes in Informatics P-21*, pages 18–30. Gesellschaft für Informatik, 2002.

- [vdA97] W.M.P. van der Aalst. Exploring the Process Dimension of Workflow Management. Computing Science Reports 97/13, Eindhoven University of Technology, September 1997.
- [vdAtH02] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. Technical Report QUT Technical report, FIT-TR-2002-06, Queensland University of Technology, Brisbane, 2002.
- [vdAvH02] Wil van der Aalst and Kees van Hee. *Workflow Management: Models, Methods, and Systems*. Cooperative Information Systems. The MIT Press, 2002.
- [WFM99] Workflow Management Coalition: Terminology & Glossary. Technical Report WFMC-TC-1011, The Workflow Management Coalition (WfMC), February 1999.
- [WK03a] Michael Weber and Ekkart Kindler. The Petri Net Kernel. In H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, editors, *Petri Net Technologies for Modeling Communication Based Systems, LNCS 2472*, pages 109–123. Springer, 2003.
- [WK03b] Michael Weber and Ekkart Kindler. The Petri Net Markup Language. In H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, editors, *Petri Net Technologies for Modeling Communication Based Systems, LNCS 2472*, pages 124–144. Springer, 2003.
- [XSch] XML Schema. URL <http://www.w3.org/XML/Schema>, April 2000. 2002-03-22.