Paper for the invited talk at EKA 2006: In E. Schnieder (ed.): Entwurf Komplexer Automatisierungssysteme, EKA 2006, 9. Fachtagung, Braunschweig, Germany, May 2006, pp. 35-55.

Concepts, Status, and Future Directions

Ekkart Kindler
Department of Computer Science
University of Paderborn
Warburger Strasse 100
D-33098 Paderborn
Germany
E-Mail: kindler@upb.de

Abstract: The *Petri Net Markup Language* (PNML) is an XML-based interchange format for Petri nets. In order to support different versions of Petri nets, its focus is on universality and flexibility, which is achieved by a technique for defining new Petri net types. For presenting and precisely defining the XML-syntax, PNML uses UML meta models: The *PNML Core Model* defines the concepts shared by all kinds of Petri nets; additional *Petri net type definitions* are UML meta models for defining the concepts that are specific to particular kinds of Petri nets. The concrete XML-syntax is then defined by mapping the concepts of these UML meta models to XML elements.

Currently, PNML is standardised as Part 2 of the International Standard ISO/IEC 15909 as the transfer syntax for three particular versions of Petri nets: *Place/Transition-Nets*, *High-level Petri Nets*, and *Symmetric Nets*.

In this paper, we outline the concepts of PNML and, in particular, of the *PNML Core Model* and its mapping to XML-syntax. In order to illustrate the definition of a particular Petri net type, we discuss the meta model for *Place/Transition-Nets*. Since the exact details of the transfer syntax for High-level nets are not yet fixed, we do not go into the details of this Petri net type. Moreover, we report on the current state of the standardisation and possible future extensions, which might be included to Part 3 of ISO/IEC 15909.

Keywords: Petri net, high-level Petri net, P/T-Net, transfer format, XML, PNML, ISO/IEC 15909

1 Introduction

Petri nets come in many versions, variants, and flavours. The *Petri Net Markup Language* (PNML) is an XML-based interchange format that supports the exchange of Petri nets among different tools. It was designed to accommodate the demands of the variety of *Petri net types*.

Currently, PNML is under standardisation as Part 2 of ISO/IEC 15909. This standard will define the *PNML Core Model* – the concepts common to all Petri nets – as well as three concrete *Petri net types*: *Place/Transition-Nets* (*P/T-Nets*), *High-level Petri Nets* (*HLPNG*), and *Symmetric Nets*[1].

Though ISO/IEC 15909-2 is not yet finalised (see [ISO05] for the latest working draft), the concepts of the *PNML Core Model* and its XML-syntax are generally agreed upon; as are the *Petri net type definition* and the XML-syntax for *P/T-Nets*. The details for HLPNGs and for *Symmetric Nets*, however, are still under discussion.

In this paper, we present the concepts of PNML. We will discuss the *PNML Core Model* and its XML-syntax as well as the technique for defining new *Petri net types*. This will be illustrated by the example of the definition of the concepts and the XML-syntax of *P/T-Nets*. Moreover, we will discuss the current state of the standardisation, the open issues, and possible future directions.

2 PNML concepts

In this section, we discuss the basic concepts of PNML and their relation. These concepts are independent from a particular XML-syntax. Therefore, PNML uses meta modelling techniques, in particular UML class diagrams, for precisely capturing these concepts. For understanding the PNML concepts, however, an in-depth understanding of these meta modelling techniques and of UML are not necessary; the UML diagrams can be considered as illustrations of the concepts defined in the text. The mapping of these concepts to XML-syntax will be defined in Sect. 3.

2.1 General principles

PNML was designed to be extensible and open for future variants of Petri nets and possibly for other use, such as the transfer of results associated with the analysis of Petri nets. In order to obtain this flexibility, the transfer format considers a Petri net as a labelled directed graph, where all type specific information of the net is represented in *labels*. *Labels* may be associated with *nodes*, *arcs*, *pages*, or with the *net* itself. This basic structure of PNML is defined in the *PNML Core Model*, which is defined in Sect. 2.2.

The *PNML Core Model* imposes no restrictions on *labels*. Therefore, the *PNML Core Model* can represent any kind of Petri net. Due to this generality of the *PNML Core Model*, there can be

---

[1] *Symmetric Nets* have been inspired by well-formed nets [CDFH91].

models that do not even correspond to a Petri net at all. For a concrete version of Petri nets, the legal *labels* will be defined by extending the *PNML Core Model* with another meta model that exactly defines the legal *labels* of this *Petri net type*; such a meta model is called a *Petri net type definition*.

Technically, the *PNML Core Model* is a UML package, and there are additional UML packages for the different *Petri net types* that extend the *PNML Core Model* package. ISO/IEC 15909-2 will define a *Petri net type* for *P/T-Nets*, for *High-level Petri Net Graphs* (HLPNGs), and for *Symmetric Nets*. HLPNGs subsume *P/T-Nets*, which means that any legal *label* of a *P/T-Net* may occur also in a HLPNG. *Symmetric Nets* are a version of HLPNGs with a restricted subset of built-in types and functions. In this paper, we concentrate on the package for *P/T-Nets* in order to illustrate the idea of defining *Petri net types*.

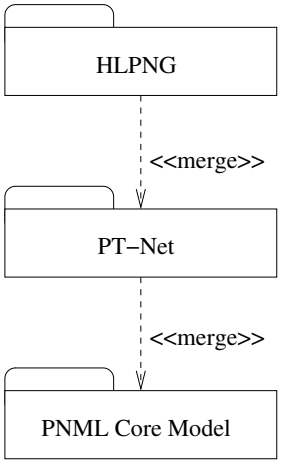

Figure 1: Overview of the UML packages of PNML

Figure 1 gives an overview of the different UML packages of PNML and of their dependencies. The *PNML Core Model* package defines the basic structure of all kinds of Petri nets; this structure will be extended by the package for the two types. The *PNML Core Model* is defined in Sect. 2.2, the *P/T-Net* package is defined in Sect. 2.3.1, and some ideas on the package for HLPNGs are discussed in Sect. 2.3.2. In Sect. 3, we show how the concepts defined in these packages are mapped to concrete XML-syntax.

2.2 PNML Core Model

Figure 2 shows the *PNML Core Model* as a UML class diagram. This diagram will be discussed in the following subsections. For an example of a Petri net model and its representation in XML, we refer to Fig. 5 and Listing 1.

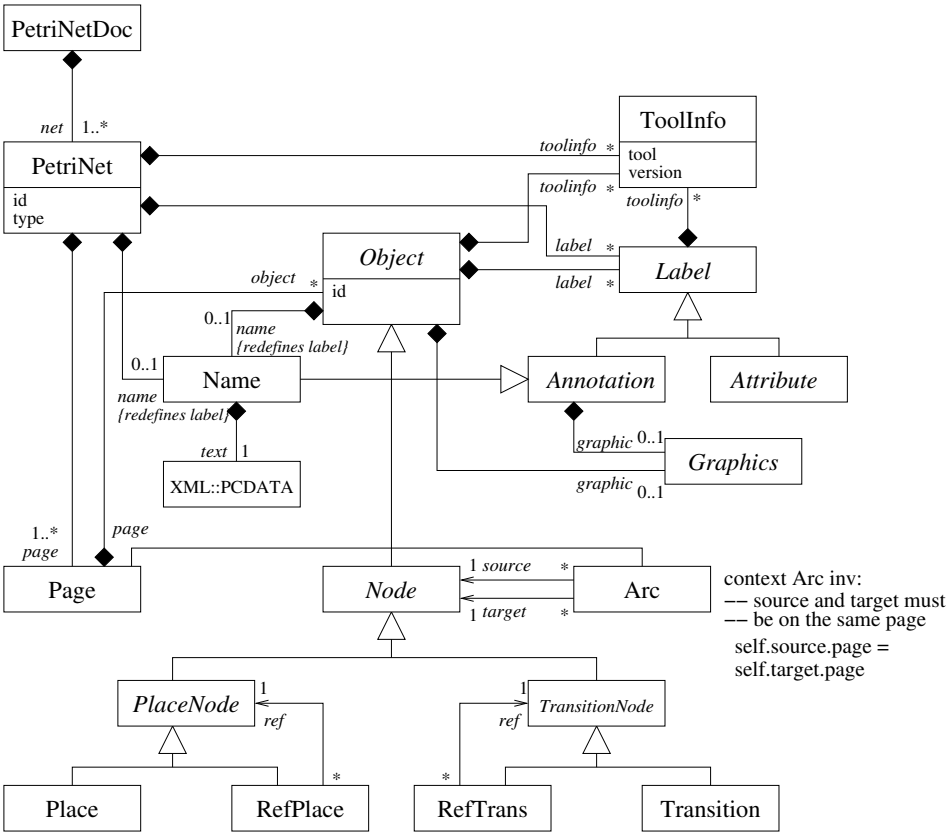

Figure 2: The PNML Core Model

2.2.1 Petri net documents, Petri nets, and objects

A document that meets the requirements of the *PNML Core Model* is called a *Petri net document* (PetriNetDoc). It may contain several *Petri nets* (PetriNet). Each *Petri net* includes a unique identifier and a type. The type is a URL referring to the name of the package with its definition.

A *Petri net* consists of some top-level *pages* that in turn consist of several *objects*. These *objects*, basically, represent the graph structure of the Petri net. Each *object* within a *Petri net document* has a unique *identifier*, which can be used for referring to this *object*. Moreover, each *object* may be equipped with *graphical information* defining its position, size, colour, shape and other information on its graphical appearance. The precise graphical information that can be provided for an *object* depends on the particular type of the *object*, which will be discussed in Sect. 2.2.4 in more detail.

An *object* of a net is a *place*, a *transition*, or an *arc*, or – as discussed below – a *page* or a *reference node*. For convenience, a *place* or a *transition* is generalised to a *node*, which can be connected by *arcs*.

Note that it is legal to have an *arc* from a *place* to a *place* or from a *transition* to a *transition* according to the *PNML Core Model*. The reason is that there are versions of Petri nets that may have such arcs. If a *Petri net type* does not support such arcs, this restriction will be imposed on it in the particular *Petri net type definition*.

2.2.2 Pages and reference nodes

Three other kinds of *objects* are used for structuring large *Petri nets*: *pages*, *reference places* (RefPlace), and *reference transitions* (RefTrans). As mentioned above, a *page* may contain *objects*; since a *page* is an *object* itself, a *page* may contain other *pages*. Therefore, there can be a hierarchy of sub-pages in a *Petri net*.

PNML requires that an *arc* must connect *nodes* on the same *page* only. The reason for this restriction is that *arcs* connecting *nodes* on different *pages* cannot be drawn graphically on a single page. In the *PNML Core Model* shown in Fig. 2, this requirement is formally expressed by the OCL expression next to the class Arc.

In order to connect *nodes* on different *pages*, a representative of one of the two *nodes* is drawn on the same *page* as the other *node*. Then, this representative may be connected with the other *node* by an *arc*. This representative is called a *reference node* because it has a reference to the node it represents. Note that a *reference place* must refer to a *place* or a *reference place* (generalised to a class PlaceNode), and a *reference transition* must refer to a *transition* or a *reference transition* (generalised to a class TransitionNode). Cyclic references among *reference nodes*, however, are not allowed.

2.2.3 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 transition condition, or an arc annotation. In addition, the *Petri net* itself and its *pages* may have some *labels*. These are called *global labels*. For example, the package for HLPNGs defines type declarations as *global labels* of a HLPNG.

PNML distinguishes two kinds of *labels*: *annotations* and *attributes*. An *annotation* comprises information that is typically displayed as text next to the corresponding *object*. Examples of *annotations* are names, initial markings, arc annotations, transition conditions, and timing or stochastic information. In contrast, an *attribute* is not displayed as text next to the corresponding *object*. Rather, an *attribute* has an effect on the shape or colour of the corresponding object. For example, an *attribute* arc type could have the domain {`normal`, `read`[2], `inhibitor`,

[2] Sometimes the read arcs are called test arcs.

`reset}`. PNML, however, does not mandate the effect of *attributes* on the graphical appearance.

Note that the classes for *label*, *annotation* and *attribute* are abstract in the *PNML Core Model*, which means that the *PNML Core Model* does not define concrete *labels*, *annotations*, and *attributes*. The only concrete *label* defined in the *PNML Core Model* is the *name*, which is a *label* that can be used with any *Petri net type*. The text of this *label* is given as XML *PCDATA*, which is expressed in the UML model by referring to the class XML::PCDATA (see Sect. 3.3). The other concrete *labels* will be defined in the packages for the concrete *Petri net type definitions* (see Sect. 2.3).

In order to support the exchange of information among tools that have different textual representations for the same concept (i.e. when they have different concrete syntax), there are two ways for representing the information within an *annotation*: *textually* in some concrete syntax and *structurally* as an abstract syntax tree (see Sect. 3.1.2 for details).

Note that *reference nodes* may have *labels*, but these *labels* do not carry any meaning. Semantically, *labels* of *reference nodes* will be ignored. They cannot specify any information about the referenced *node*, since this *node* has its own *labels*. This choice simplifies the definition of the semantics of *Petri nets* with *pages* by simply removing all *pages* and *reference nodes* and connecting the referenced *nodes* by arcs directly. This is called *flattening* [KW01, WK03].

2.2.4 Graphical information

Graphical information can be associated with each *object* and each *annotation*. 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 *object's annotation*, it includes its relative position with respect to the *object* it is attached to; for an *annotation* of a *page*, the position is absolute. There can be further *graphical information* concerning the size, colour and shape of *nodes* or *arcs*, or concerning the colour, font and size of *labels* (see Sect. 3.1.3 for more information).

2.2.5 Tool specific information

For some tools, it might be necessary to store tool specific information, which is not meant to be used by other tools. In order to store this information, *tool specific information* (ToolInfo) may be associated with each *object* and each *label*. Its internal structure 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 adding this information. Therefore, other tools can easily ignore it, and adding *tool specific information* will never compromise a *Petri net document*.

The same *object* may be tagged with *tool specific information* from different tools. This way, the same document can be used and changed by different tools at the same time. The intention is that a tool should not change or delete the information added by another tool as long as the corresponding *object* is not deleted. Moreover, *tool specific information* should be self contained and should not refer to other *objects* of the net because the deletion of other *objects* by a different tool might make this reference invalid and leave the *tool specific information* inconsistent. This use of *tool specific information* is strongly recommended; however, it is not normative!

2.3 Petri net type definitions

Next, we discuss the meta models for two versions of Petri nets: *P/T-Nets* and HLPNGs. These meta models define the *labels* of the respective *Petri net type*.

Though HLPNGs conceptually generalise *P/T-Nets*, there are differences in syntax. Of course, there are mappings from the syntax of the concepts of *P/T-Nets* to HLPNGs. But, in order not to force tools that support *P/T-Nets* only to use the more complicated and verbose syntax of HLPNGs, we introduce a simple syntax for *P/T-Nets*. The transfer syntax for HLPNGs, however, also includes the syntax of *P/T-Nets* (cf. Fig. 1).

Note that *Symmetric Nets*, which are defined in an Addendum to Part 1 of ISO/IEC 15909, are a special version of HLPNGs with a restricted set of built-in types and operations. This is a semantical restriction which is imposed in addition to the restrictions of HLPNGs, which will not be discussed in this paper.

2.3.1 Place/Transition-Nets

This section defines the meta model for *P/T-Nets* in terms of a UML package *PT-Net*. Note that the UML meta models do not fully specify the XML-syntax for these concepts. The exact mapping to XML-syntax will be defined in Sect. 3.2.

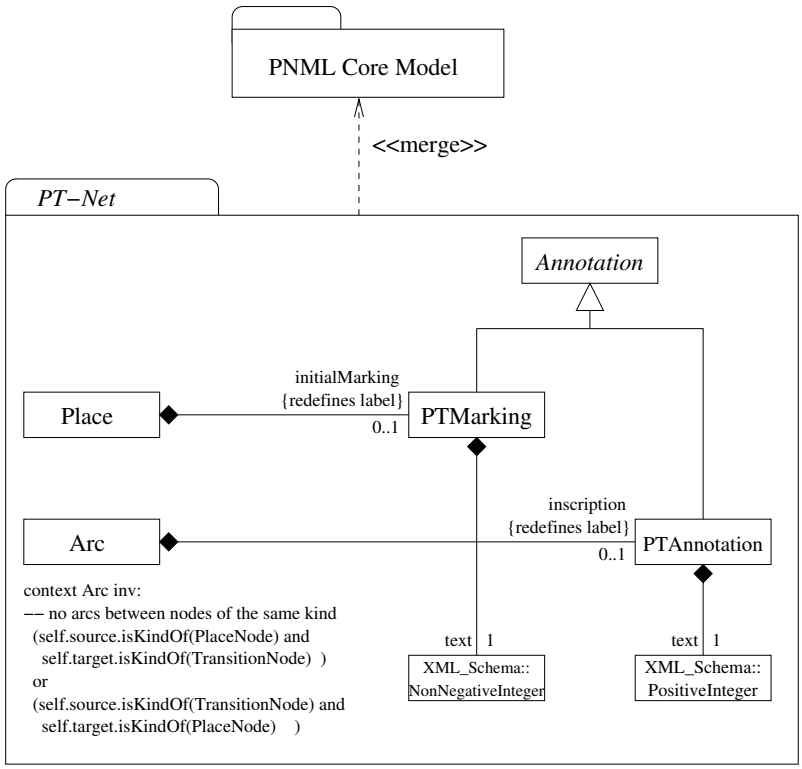

Figure 3: The package *PT-Net*

A *P/T-Net* is a Petri net, where each *place* is *labelled* with a natural number representing the *initial marking*, and each *arc* is *labelled* with a non-zero natural number representing the *arc annotation*. Figure 3 shows the package *PT-Net*. This model defines that each *place* can have an *annotation PTMarking* that represents a natural number. Technically, the type and the representation of the contents of this *label* is defined by referring to the data type *NonNegativeInteger* of XMLSchema [SMe00]. If this *label* is missing, it is assumed to be 0. Each *arc* can have an *annotation PTAnnotation* that consists of a non-zero natural number, where the representation of this *label* is defined by the data type *PositiveInteger* of XMLSchema. If this *label* is missing, it is assumed to be 1. The OCL expression on the lower left side of Fig. 3 expresses that, in *P/T-Nets*, an *arc* must not connect a *place* to a *place* or a *transition* to a *transition*.

Note that the only classes defined here are *PTMarking* and *PTAnnotation*. The classes Place, Arc, and Annotation come from the *PNML Core Model* package. They are imported here in order to define the concrete *labels* attached to these *nodes* in this particular *Petri net type*, i.e. in *P/T-Nets*. Likewise, the classed prefixed with XML and XMLSchema are defined in other packages in order to refer to standard concepts of XML and standard data types of XMLSchema (see Sect. 3.3).

2.3.2 High-Level Petri Net Graphs

This section discusses the basic concepts for HLPNGs. Note that this is still under discussion and subject to change. Therefore, we do not discuss the exact XML-syntax for HLPNGs in this paper.

A HLPNG is a Petri net that is equipped with *declarations* that define *types*, *functions*, and *variables* that are the basis for constructing *expressions*. The *declarations* can be *annotations* of the *Petri net* itself or of some of its *pages*; the *types* can be *annotations* of *places*; and a *place* can also have an *annotation* with its *initial marking* which is an *expression* for some multiset; an *arc annotation* is some expression; a *transition* can have a *condition* as its *annotation*, which is an expression of type boolean.

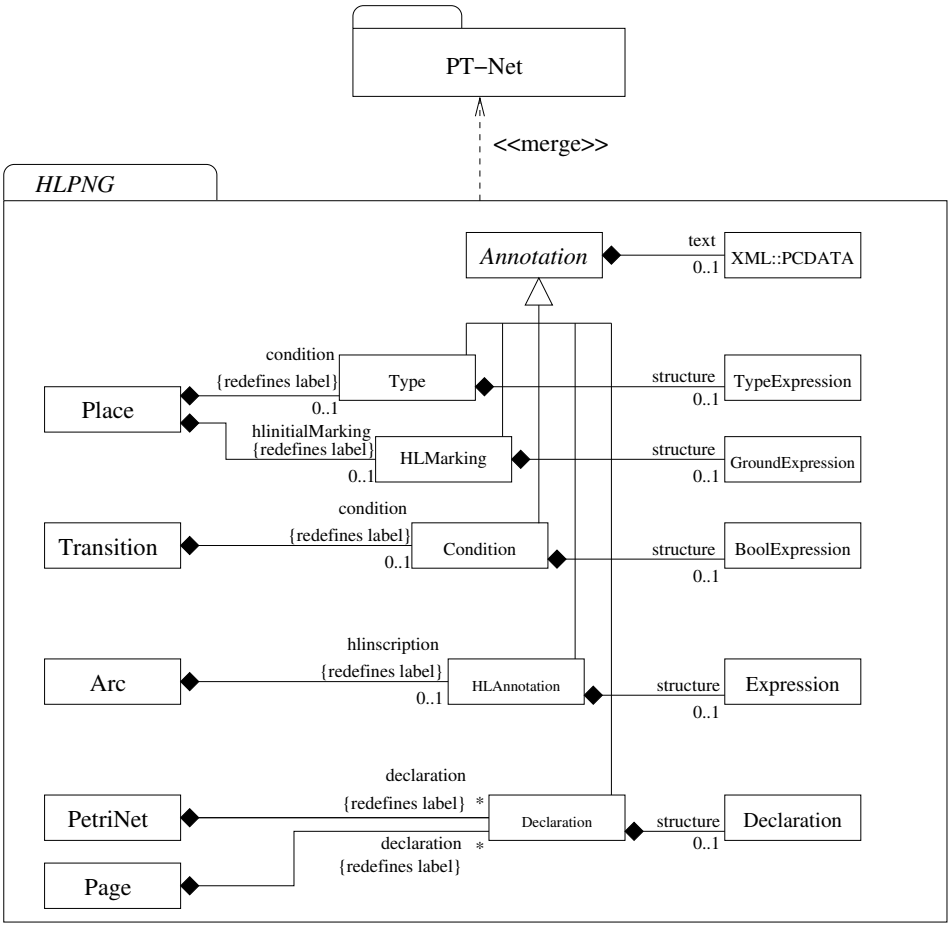

Figure 4: The Package *HLPNG*

The basic structure of a package defining the concepts of HLPNGs is outlined in Fig. 4. The exact meta model for all these concepts is still under discussion. Eventually, these concepts will be mapped to some MathML elements.

Note that this meta model will define an abstract syntax for the HLPNG concepts only. In order to allow tools to store some concrete text representing these concepts, the *annotation* may also consist of *text*, which should be the same expression in the concrete syntax of some tool. The concrete syntax, however, is not defined in PNML. For properly exchanging HLPNGs it will be necessary to have the abstract syntax for all HLPNG *labels*. This will give rise to two levels of conformance to HLPNGs: the *textual conformance* does not require that the structural information is there, whereas the *structural conformance* requires that there is the structural information for all HLPNG *labels* (see Sect. 4.1 for more details).

In addition to the *annotations* defined above, the package for HLPNGs also requires that *arcs* must not connect two *places* and must not connect two *transitions*; this however, is a property inherited from *P/T-Nets*. Therefore, it is not necessary to include this condition here again.

3 PNML syntax

In Sect. 2, we have defined the concepts of the *PNML Core Model* and the concepts of *P/T-Nets*. We did not define the precise XML-syntax for representing these concepts. The precise XML-syntax for the *PNML Core Model* and for *P/T-Nets* will be defined below by mapping the concepts defined in the meta models to XML elements.

Section 3.1 defines the mapping of the *PNML Core Model* to XML. Section 3.2 defines the mapping of *P/T-Nets* to XML. In fact, we give general rules, how the concepts of a package defining some *Petri net type* are mapped to XML-syntax.

The final version of ISO/IEC 15909-2 will include a RELAX grammar [ClMu] defining the precise XML-syntax of PNML and the different versions of Petri nets. Here, we will not give this RELAX grammar because it is not yet finalised. A preliminary version can be found on the PNML web pages and in the current working draft of ISO/IEC 15909-2 [PNML, ISO05].

3.1 PNML Documents

The mapping of the *PNML Core Model* concepts to XML-syntax is defined for each class of the *PNML Core Model* package.

3.1.1 PNML elements

Each concrete class[3] of the *PNML Core Model* is mapped to an XML element. The mapping of these classes along with the attributes and their data types is given in Table 1. These XML elements are the *keywords* of PNML and are called *PNML elements* for short. For each *PNML*

[3] A class in a UML diagram is concrete if its name is not displayed in italics.

Table 1: Mapping of the *PNML Core Model* concepts to *PNML elements*

Class	XML element	XML attributes
PetriNet Doc	`<pnml>`	
PetriNet	`<net>`	`id:ID` `type:anyURL`
Place	`<place>`	`id:ID`
Transition	`<transition>`	`id:ID`
Arc	`<arc>`	`id:ID` `source:IDRef` (Node) `target:IDRef` (Node)
Page	`<page>`	`id:ID`
RefPlace	`<referencePlace>`	`id:ID` `ref:IDRef` (Place or RefPlace)
RefTrans	`<referenceTransition>`	`id:ID` `ref:IDRef` (Transition or RefTrans)
ToolInfo	`<toolspecific>`	`tool:string` `version:string`
Graphics	`<graphics>`	

element, the UML compositions of the *PNML Core Model* define in which elements it may occur as a child element.

The data type ID in Table 1 refers to a set of unique identifiers within the *PNML Document*. The data type IDRef refers to the set of all identifiers occurring in the document, i.e. they are meant as references to identifiers. A reference at some particular position, however, is restricted to objects of a particular type – as defined in the *PNML Core Model*. For instance, the attribute `ref` of a *reference place* must refer to a *place* or to a reference place of the same net. The set to which a reference is restricted, is indicated in the table (e.g. for a *reference place*, the attribute `ref` should refer to the `id` of a *place* or a *reference place*).

3.1.2 Labels

Except for *names*, there are no *PNML elements* for *labels* because the *PNML Core Model* does not define any other concrete *labels*. For concrete *Petri net types*, such as *P/T-Nets* and

HLPNGs, the corresponding packages define these labels. The mapping of the *name* label of the *PNML Core Model* will follow the general rules for mapping *labels* to XML as discussed below and in Sect. 3.2. The *name* will be included in the XML element <name>.

In *PNML Documents*, any XML element that is not defined in the *PNML Core Model* (i.e. not occurring in Table 1) is considered as a *label* of the *PNML element* in which it occurs. For example, an <initialMarking> element could be a *label* of a *place*, which represents its initial marking. Likewise, <inscription> could represent the arc inscription.

A legal XML element for a *label* must contain at least one of the two following elements, which represents the actual value of the *label*: a <text> element represents the value of the *label* as a simple string; the <structure> element can be used for representing the value as an abstract syntax tree.

An optional *PNML element* <graphics> defines its graphical appearance; and optional *PNML elements* <toolspecific> may add *tool specific information* to the *label*.

3.1.3 Graphics

All *objects* and all *labels* may be equipped with *graphical information*. The internal structure of the corresponding PNML <graphics> element, i.e. the legal XML children, depends on the element in which the graphics element occurs. Table 2 shows the elements which may occur within the <graphics> element, all of which are optional.

The <position> element defines the absolute position of a *node*, whereas the <offset> element defines the relative position of an *annotation*. Each absolute or relative position refers to Cartesian coordinates (*x, y*). As for many graphical tools, the *x*-axis runs from left to right and the *y*-axis from top to bottom. The *reference point* of a *node* is its middle. For an *arc*, the (possibly empty) sequence of <position> elements defines its intermediate points (bend points).

Table 2: Possible child elements of the <graphics> element

Parent element class	**Sub-elements of <graphics>**
Node, Page	<position> <dimension> <fill> <line>
Arc	<position> (zero or more) <line>
Annotation	<offset> <fill> <line> <font>

Table 3 defines the *attributes* for each graphical element defined in Table 2. The domain of the attributes refers to the data types of either XML Schema [SMe00], or Cascading Stylesheets 2 (CSS2) [BLLe98], or is given by an explicit enumeration of the legal values.

The <position> element defines the absolute position of a *node* on its *page*. The <offset> element defines the position of an *annotation* relative to the position of the *object* – for a *global annotation*, it defines the absolute position on that *page*.

For an *arc*, the (possibly empty) list of <position> elements define intermediate points (bend points) of the *arc* – the start and end point, however, are not given explicitly since these points come from the source and the target *node* of the *arc*. Altogether, the *arc* is a path from the source *node* of the *arc* to the target *node* of the *arc* via the intermediate points. Depending on the value of the attribute shape of element <line>, the path is displayed as a broken line or as a (quadratic) Bezier curve, where the points act as line connectors or Bezier control points.

The <dimension> element defines the height and the width of a *node*. Depending on the ratio of height and width, a *place* is displayed as an ellipse rather than a circle. A *transition* is displayed as a rectangle of the corresponding size. If the dimension of an element is missing, each tool is free to use its own default value for the dimensions.

The two elements <fill> and <line> define the interior and outline colours of the corresponding element. The value assigned to a colour attribute must be a RGB value or a predefined colour as defined by CSS2. When the attribute gradient-color is defined, the fill colour continuously varies from color to gradient-color. The additional attribute gradient-rotation defines the orientation of the gradient. If the attribute image is defined, the node is

Table 3: PNML graphical elements

XML element	Attribute	Domain
<position>	x	decimal
	y	decimal
<offset>	x	decimal
	y	decimal
<dimension>	x	nonNegativeDecimal
	y	nonNegativeDecimal
<fill>	color	CSS2-color
	image	anyURI
	gradient-color	CSS2-color
	gradient-rotation	{vertical, horizontal, diagonal}
<line>	shape	{line, curve}
	color	CSS2-color
	width	nonNegativeDecimal
	style	{solid, dash, dot}
<font>	family	CSS2-font-family
	style	CSS2-font-style
	weight	CSS2-font-weight
	size	CSS2-font-size
	decoration	{underline, overline, line-through}
	align	{left, center, right}
	rotation	decimal

displayed as the image at the specified URI, which must be a graphics file in JPEG or PNG format. In this case, all other attributes of <fill> and <line> are ignored.

For an *annotation*, the <font> element defines the font used to display the text of the *label*. The complete description of possible values of the different attributes can be found in the CSS2 specification [BLLe98]. Additionally, the align attribute defines the justification of the text with respect to the *label's* coordinates. Depending on the value of this attribute, the reference point of the *label* is the lower left or right corner or the center of the *label*; the default is the lower left corner. The rotation attribute defines a clockwise rotation of the text.

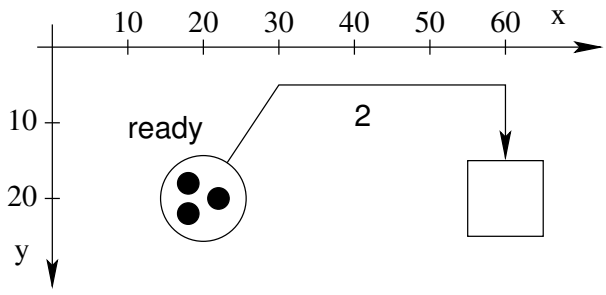

Figure 5: A simple *Place/Transition-Net*

3.1.4 Example

In order to illustrate the structure of a *PNML Document*, we give an example representing the Petri net shown in Fig. 5, which actually is a *P/T-Net*. Listing 1 shows the corresponding *PNML Document*. It is a straight-forward translation, where there are *labels* for the *names* of *objects*, for the *initial markings*, and for *arc annotations*.

Note that a tool would typically display the initial marking as a textual label. But, we assume that this net is viewed by an imaginary tool *PN4all*; in the *PNML Document*, this tool has added some *tool specific information* in the *annotation* for the *initial marking*. We assume that the imaginary tool interprets its own *tool specific information* in such a way that it shows the tokens at individual positions as given in the elements `<tokenposition>`

Since there is no information on the dimensions in this example (in order to fit the listing on a single page), the tool has chosen its default dimensions for the *place* and the *transition*.

Listing 1: PNML code of the example net in Fig. 5

```
<pnml xmlns="http://www.example.org/pnml">
  <net id="n1" type="http://www.example.org/pnml/PTNet">
    <page id="top-level">
      <name>
        <text>An example P/T-net</text>
      </name>
      <place id="p1">
        <graphics>
          <position x="20" y="20"/>
        </graphics>
        <name>
          <text>ready</text>
          <graphics>
            <offset x="-10" y="-8"/>
          </graphics>
        </name>
        <initialMarking>
          <text>3</text>
          <toolspecific tool="PN4all" version="0.1">
            <tokenposition x="-2" y="-2" />
            <tokenposition x="2" y="0" />
            <tokenposition x="-2" y="2" />
          </toolspecific>
        </initialMarking>
      </place>
      <transition id="t1">
        <graphics>
          <position x="60" y="20"/>
        </graphics>
      </transition>
      <arc id="a1" source="p1" target="t1">
        <graphics>
          <position x="30" y="5"/>
          <position x="60" y="5"/>
        </graphics>
        <inscription>
          <text>2</text>
          <graphics>
            <offset x="15" y="-2"/>
          </graphics>
        </inscription>
      </arc>
    </page>
  </net>
</pnml>
```

3.2 Mapping a Petri net type definition to XML

Based on the *PNML Core Model* of Section 2.2, Sections 2.3.1 and 2.3.2 define the concepts of two particular *Petri net types*, which restrict *PNML Documents* to the particular *labels* as defined in the corresponding packages.

These packages define the *labels* that can or must be used in the particular *Petri net type*. Here, we will show how to map such a package to the corresponding XML-syntax. If not stated otherwise, this mapping is according to the same rules for each *Petri type definition*. We will explain these rules by the help of the example of *P/T-Nets* (see Fig. 3 and the *PNML Document* in Listing 1).

The *P/T-Net* package defines two new *labels* that can be used in a *P/T-Net*: *PTMarkings* and *PTAnnotations*. Each place can have an annotation *PTMarking*, and each *arc* can have an *annotation PTAnnotation*. This is indicated by the compositions in the UML diagram.

The XML-syntax for these *labels* is derived from the roles of these compositions. For example, an *annotation PTMarking* is mapped to an XML element `<initialMarking>`, and an *annotation PTAnnotation* is mapped to an element `<inscription>` (see Listing 1).

Since all *labels* in this package are derived from *annotation*, all graphical elements defined for *annotations* may occur as children in these elements. In the *P/T-Net* package, each *label* consists of a `<text>` element, which defines the actual content of this annotation. For *P/T-Nets* we do not need the structured element. The corresponding classes define the XML content of the `<text>` element as defined by the references to external definitions.

3.3 External definitions

In the above definitions, we refer to XML or XML Schema for defining the internal structure of some *labels*. For example, the *name* label refers to XML's PCDATA; other *labels* refer to data types of XML Schema. In order to refer to external XML technologies, we use the notation `XML::PCDATA` or `XMLSchma::NonNegativeNumber` in the corresponding classes of the UML meta models.

4 Conformance, status, and future directions

In the previous sections, we have discussed the concepts and the XML-syntax of PNML. In this section, we discuss what will or could be in the forthcoming International Standard ISO/IEC 15909-2. Note that this standard is, basically, interested in the transfer syntax for HLPNGs, but it also deals with the general structure of Petri nets, with P/T-Nets, and with a restricted version of HLPNGs, called *Symmetric Nets*.

4.1 Conformance

From the structure of PNML, there are different levels of conformance to PNML. The first level is a general conformance to the *PNML Core Model*, which does not impose any restrictions on the *Petri net type* or on the *labels* attached to it. An XML document is conformant to the *PNML Core Model* if it meets the definitions of Sect. 2.2 (*PNML Core Model*) and 3.1 (its XML representation). Such a document is called a *PNML Document* or a *Petri Net Document*.

The other levels of conformance concern the different *Petri Net Types*. A *PNML Document* is a conformant *Place/Transition-Net*, if it meets the additional restrictions of Sect. 2.3.1 (*P/T-Net Model*) and Sect. 3.2 (mapping to XML). Such a document is called a *PNML Place/Transition Net Document*.

For *High-level Petri Nets*, there are two different levels of conformance. The first level is conformance on the textual level, which basically ignores the meaning of the high-level inscriptions. This level does not require that there are structural elements in the high-level *labels*. A *PNML Document* is a *textually conformant High-level Petri Net*, if it meets the additional restrictions of Sect. 2.3.2. Such a document is called a *textually conformant PNML High-level Petri Net Document*. If there are the abstract syntax trees for all high-level inscriptions, and these inscriptions are syntactically correct, the *PNML Document* is a *structurally conformant High-level Petri Net Document*.

A *structurally conformant High-level Document* is a *conformant Symmetric Net Document* if, in addition, the structural inscriptions use the built-in sorts, functions and constants of *Symmetric Nets* only.

4.2 Status

Note that ISO/IEC 15909-2 is still under discussion. Once it will be finished, it will be PNML version 2.0. From the latest discussions in the editorial meetings, it appears that the concepts and the XML-syntax of the *PNML Core Model* and the type definition for *P/T-Nets* will not change any more.

For HLPNGs as well as for *Symmetric Nets*, however, there are still open issues and a final version was not yet agreed on. This is the reason for not including details on these types here. The publication of a reliable draft for this part of the standard is scheduled for June 2006. There will be a meta model for HLPNGs, which was first outlined in WD 0.5.0 of ISO/IEC 15909-2; the concrete XML-syntax will map these concepts to MathML.

In addition to the meta models defining the *PNML Core Model* and the different types, the final version of ISO/IEC 15909-2 will also contain a RELAX grammar for precisely defining the XML-syntax, which should help to easily validate *PNML Documents*. A preliminary version of

this RELAX grammar can be found in [ISO05], where the part for the *PNML Core Model* needs some minor polishing, the part for HLPNGs, however, needs major revisions. Note that some restrictions of PNML cannot be captured by current XML technologies and will not be covered by the RELAX grammar. Still, these restrictions are normative for conformant *PNML Documents*.

ISO/IEC 15909-2 will probably also contain two informative annexes. One annex will deal with the graphical appearance of *PNML Documents*, which will be an XSLT transformation of *PNML Documents* to SVG; this is based on the ideas of Stehno [Ste02]. The other will deal with a framework and an API for the implementation of readers and writers for *PNML Documents*, which is an implementation provided by Hillah et al. [HKPT05].

4.3 Future directions

Though the presentation of PNML in ISO/IEC 15909-2 is structured in the *PNML Core Model* and the models for the different *Petri net types*, the formalisation of the concept of a *Petri net type definition* was excluded from Part 2. This technique is subject to Part 3 of ISO/IEC 15909, which is currently planned.

In Part 3, the concept and the technology for defining *Petri net types* will be explicitly defined. Moreover, Part 3 will define more features of Petri nets such as read arcs, inhibitor arcs, and reset arcs as well as timing concepts and concepts from stochastic Petri nets. Based on these concepts, Part 3 will standardise further versions of Petri nets. Also the API framework for reading and writing *PNML Documents* of a particular type could be included to this part of the standard. Moreover, Part 3 might cover the definition and use of *Petri net modules* as proposed in [KW01, WK03].

5 Conclusion

In this paper, we have outlined the concepts of the upcoming standard on a transfer format for Petri nets: ISO/IEC 15909-2. Once finalised it will be PNML version 2.0 as compared to earlier versions of PNML presented in scientific papers [JKW00a, WK03, BCvH$^+$03]. In this presentation, we concentrated on the part of PNML that seems to be agreed upon, which is the *PNML Core Model* and *P/T-Nets*. In particular, the definition of *P/T-Nets* illustrates how new *Petri net types* could be defined in the future, which will be subject to Part 3 of ISO/IEC 15909.

Actually, the focus of ISO/IEC 15909-2 is on the transfer syntax for high-level Petri nets. Here, we could only sketch the basic idea of the XML-syntax for HLPNGs because the concepts and, more importantly, the representation in MathML are still under discussion. The PNML home page [PNML] will inform on new developments and the ongoing discussions, which should be finalised by June 2006. The formal standardisation process, however, will take some more time.

Acknowledgements

I would like to thank all people who work on ISO/IEC 15909-2 and those who have contributed to or commented on earlier versions of PNML, which are (in alphabetical order): Jonathan Billington, Søren Christensen, Jörg Desel, Erik Fischer, Giuliana Franceschinis, Kees van Hee, Lom Hillah, Nisse Husberg, Matthias Jügel, Albert Koelmans, Fabrice Kordon, Olaf Kummer, Kjeld Høyer Mortensen, Laure Petrucci, Renier Post, Wolfgang Reisig, Stefan Roch, Karsten Schmidt (Wolf), Christian Stehno, Nicolas Trèves, Kimmo Varpaaniemi, Michael Weber, and Lisa Wells.

6 Bibliography

[BBK+00] R. Bastide, J. Billington, E. Kindler, F. Kordon, and K.H. Mortensen (editors). *Meeting on XML/SGML based Interchange Formats for Petri Nets*, Århus, Denmark, June 2000.

[BCvH+03] J. Billington, S. Christensen, K. van Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, and M. 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, 24th International Conference*, *LNCS* 2679, pages 483–505. Springer, June 2003.

[BLLe98] B. Bos, H. W. Lie, C. Lilley, and I. Jacobs (editors). Cascading Style Sheets, level 2 – CSS2 Specification. 1998. URL `http://www.w3.org/TR/CSS2`.

[CDFH91] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. On Well-Formed Coloured Nets and their Symbolic Reachability Graph. In K. Jensen and G. Rozenberg (editors), *Highlevel Petri Nets: Theory and Application*, pages 373–39. Springer 1991.

[ClMu] J. Clark and M. Murata (editors). RELAX NG specification. URL `http://www.oasis-open.org/committees/relax-ng/`.

[Cla99] J. Clark (editor). XSL Transformations (XSLT) Version 1.0. 1999. URL `http://www.w3.org/TR/XSLT/xslt.html`.

[FJe03] J. Ferraiolo, F. Jun, and D. Jackson (editors). Scalable Vector Graphics (SVG) 1.1 Specification. 2003. URL `http://www.w3.org/TR/SVG11/`.

[HKPT05] L. Hillah, F. Kordon, L. Petrucci, and N. Trèves. Model engineeing on Petri nets for ISO/IEC 15909-2: API framework for Petri net types metamodels. *Petri Net Newsletter*, 69:22–40, 2005.

[ISO04] ISO/IEC 15909-1. Software and System Engineering – High-Level Petri Nets – Part 1: Concepts, Definitions and Graphical Notation. December 2004.

[ISO05] ISO/IEC 15909-2 (Working Draft 0.9.0). Software and System Engineering – High-Level Petri Nets – Part 2: Transfer Format. June 2005. URL `http://www.upb.de/cs/kindler/publications/copies/ISO-IEC15909-2-WD0.9.0.Ballot.pdf`.

[JKW00a] M. Jüngel, E. Kindler, and M. Weber. The Petri Net Markup Language. *Petri Net Newsletter*, 59:24–29, 2000.

[JKW00b] M. Jüngel, E. Kindler, and M. Weber. The Petri Net Markup Language. In S. Philippi (editor), *7.Workshop Algorithmen und Werkzeuge für Petrinetze, AWPN*, pages 47–52, Universität Koblenz-Landau, Germany, June 2000.

[KP04] F. Kordon and L. Petrucci. Structure of abstract syntax trees for coloured nets in PNML. Version 0.2 (draft), June 2004. URL `http://mefosyloma.cnam.fr/PDF/prop-tree-struc.pdf`.

[KW01] E. Kindler and M. Weber. A universal module concept for Petri nets. An implementation-oriented approach. Informatik-Berichte 150, Humboldt-Universität zu Berlin, June 2001.

[PNML] The Petri Net Markup Language Home Page. URL `http://www.informatik.hu-berlin.de/top/pnml/`.

[SMe00] M. Sperberg-McQueen and H. Thompson (editors). XML Schema. April 2000. URL `http://www.w3.org/XML/Schema`.

[Ste02] C. 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.

[Ste05] C. Stehno. Interchangable high-level Petri nets. *Petri Net Newsletter*, 69:8–21, 2005.

[WK03] M. Weber and E. Kindler. The Petri Net Markup Language. In H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber (editors), *Petri Net Technology for Communication Based Systems*, *LNCS* 2472, pages 124–144. Springer 2003.