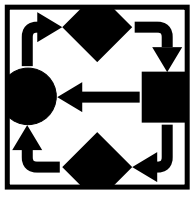
Renew – XML Format Guide

Olaf Kummer
Frank Wienberg
Michael Duvigneau

University of Hamburg
Department for Informatics
Theoretical Foundations Group
Distributed Systems Group

Release 1.5.2
July 3, 2001



Arbeitsbereich TGI
— Renew —
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
D-22527 Hamburg
Germany

This document was prepared using the LaTeX typesetting system.
This document is contained in the files `doc/xmlformat.dvi`, `doc/xmlformat.ps`, and `doc/xmlformat.pdf` as distributed together with Renew 1.5.2.

Chapter 1

The XML File Format of Renew

This document describes the file format that Renew uses for exports and imports in XML (extensible mark-up language). It is based on a paper presented at the Meeting on XML/SGML based Interchange Formats for Petri Nets [1].

The main section consists of a commented DTD (document type definition). Due to the format's simplicity it was possible to complete the implementation in one week's amount of work by a single person.

Renew itself uses an intermediate layer called shadow nets for the representation of net data. This layer formed the basis for the file format, which was then adapted to include graphical information. It is expected that with certain modifications this format can be used for different tools, too.

1.1 The File Format

The DTD is given in plain ASCII text, no special characters are used. Most keywords of the Petri net file format are given in lower case. Tags and attribute names in XML are case sensitive. In order to be consistent, it was decided to extend case-sensitivity even to those keywords that appear within attribute values.

```
<?xml encoding="US-ASCII"?>
```

The top-level element of every file is the `net`. It is conceivable to store multiple nets in one file, which would lead to an additional element `netsystem`. Currently, this is not needed.

Within the single element `net` the net elements are positioned in such an order that no forward references occur in the common cases. This simplifies parsing, which is difficult enough, and puts only a small burden on the XML generation code.

```
<!ELEMENT net (place*, transition*, arc*, annotation*)>
```

Nets are characterized by an ID, and a type. The ID is suggested for future extensions and for consistency, because all other net elements will contain an ID, too. Currently, the ID is simply `N`. If explicit IDs are given, it is supposed to consist of the letter `N` and a decimal integer in the range 1 to $2^{31} - 1$.

The type attribute is supposed to differentiate between different net formalisms. Possible values might be `ptnet` for P/T-nets or `hlnet` for high-level nets. A more fine-grained distinction might be required.

```
<!ATTLIST net
  id ID #REQUIRED
  type CDATA #IMPLIED>
```

Places possess an ID that may be referenced by arcs or other net constituents that require topological information. A place ID consists of the letter `I` followed by a decimal integer in the range 1 to $2^{31} - 1$. It is required to start the ID with a letter, because this is enforced for every attribute of the type `ID` by the XML standard. Because Renew uses numbers for IDs internally, only numeric characters may follow.

A place type might be given. Currently, only the type `ordinary` is used, but other types like `fifo` for FIFO-places might be added easily. Within the `place` element an optional graphics specification might be nested and an arbitrary number of annotations might be given.

```
<!ELEMENT place (graphics?, annotation*)>
<!ATTLIST place
  id ID #REQUIRED
  type CDATA #IMPLIED>
```

Transitions are handled exactly like places. The type of a transition is also `ordinary` in all cases.

```
<!ELEMENT transition (graphics?, annotation*)>
<!ATTLIST transition
  id ID #REQUIRED
  type CDATA #IMPLIED>
```

The IDs are assembled in the same way as place IDs. Transition IDs and place IDs and in fact all IDs that will be subsequently described share the same number space, i.e., no ID may be used more than once.

Only after all net nodes have been read in, arcs are listed in the file. Besides their own ID, arcs reference the IDs of their source and target nodes. Exactly one of `source` and `target` must reference a place, the other attribute must reference a transition.

Arc can belong to several types. The `type` attribute may have one of the values `ordinary`, `double` (an arc that reserves a token during the transition's firings), `test` (an arc that checks for the presence of a token without removing it), `multi-ordinary` (an ordinary arc that can carry a multiset of tokens), `inhibitor` (an arc that verifies the absence of a certain token), and `clear` (an arc that removes all tokens in a place).

```
<!ELEMENT arc (graphics?, annotation*)>
<!ATTLIST arc
  id ID #REQUIRED
  source IDREF #REQUIRED
  target IDREF #REQUIRED
  type CDATA #IMPLIED>
```

Nets, places, arcs, and transitions may be complemented with annotations. Annotations consist of textual information that specifies the precise behaviour of a net or a net element. Annotations may carry information on their graphical representation. It is required that they give their actual text in a separate `<text>` element. This element was introduced to avoid elements with mixed content, i.e. elements that contain text and subelements at the same time. Because the annotation text is potentially very long, is was not appropriate to use an attribute either.

```
<!ELEMENT annotation (text, graphics?)>
<!ATTLIST annotation
  id ID #REQUIRED
  type CDATA #IMPLIED>
<!ELEMENT text (#PCDATA)>
```

The possible types of an annotation depend of the enclosing element of the annotation. Currently, nets may be annotated with the types `comment`, `name` and `declaration`, where the last type is used for variable declarations and import statements.

Places may be annotated by the types `comment`, `name` (the place's name), `initialmarking` (a number of expressions that are used to get the place's initial marking), `currentmarking` (the same for a current marking, currently not implemented), `capacity` (the maximum amount of tokens that may reside in a place), and `type` (a type that restricts the tokens that are valid for this place). At the moment, capacities and current marking annotations are not supported by the tool.

Valid transition annotation types include `comment`, `name` (the transition's name), `guard` (an expression that must evaluate to true before the transition is enabled), `expression` (a formula that must hold before a transition is activated), `action` (a code segment that is evaluated during the transitions firing), `uplink` and `downlink` (for the specification of synchronous channels, specific to reference nets).

Besides `comment`, arcs allow the annotation type `expression` to specify an expression that is evaluated to obtain the token moved by this arcs. In the case of P/T-nets, the expression must be an integer constant that denotes the number of tokens moved.

If the annotation type is omitted, it has to be inferred from the annotation text. Types that are not recognized by the parser should be reported to the user, but also inferred if possible.

This completes the description of the semantic elements. All graphical information is bundled in a single element. The rationale for an individual graphics element is that non-graphical applications can easily ignore any graphics-related information this way. It would be more difficult, if all attributes were listed directly within the net elements.

All coordinates and sizes are expected in pixel units. All numeric values are given in the form of Java number literals. All sizes default to application dependent values. An offset describes the offset of the enclosing net element's center to the net element's center. If an element occurs directly inside the `net` like places or transitions, the offset is relative to the index origin of the net, which lies at the upper left corner. Positive x-coordinates run right, positive y-coordinates run down.

```
<!ELEMENT graphics (size?, textsize?, offset?,
  fillcolor?, pencolor?, textcolor?, point*, data*)>
```

A graphics element can contain many individual elements for various units of information.

```
<!ELEMENT size EMPTY>
<!ATTLIST size
  w CDATA #REQUIRED
  h CDATA #REQUIRED>
<!ELEMENT textsize EMPTY>
<!ATTLIST textsize
  size CDATA #REQUIRED>
<!ELEMENT offset EMPTY>
<!ATTLIST offset
  x CDATA #REQUIRED
  y CDATA #REQUIRED>
```

In the case of colors, each color elements contains exactly one subelement. Usually this is a subelement that lists the read, green, and blue component separately, but it can also be one of two special cases `transparent` and `background`. These elements request that the element should be drawn not at all or in the background color, thereby making it invisible, but possibly obscuring other elements. The possibility of completely unrelated values for these attributes was the reason why the red, green, and blue values were not directly included in the color elements.

```
<!ELEMENT fillcolor (RGBcolor | transparent | background)>
<!ELEMENT pencolor (RGBcolor | transparent | background)>
<!ELEMENT textcolor (RGBcolor | transparent | background)>
<!ELEMENT RGBcolor EMPTY>
<!ATTLIST RGBcolor
  r CDATA #REQUIRED
  g CDATA #REQUIRED
  b CDATA #REQUIRED>
<!ELEMENT transparent EMPTY>
<!ELEMENT background EMPTY>
```

A graphics element may contain an arbitrary number of points that specify the exact geometry. The interpretation of the points depends on the enclosing object. Points are given in coordinates relative to the net. Currently, points are used to denote intermediate points of arcs.

```
<!ELEMENT point (x,y)>
<!ATTLIST point
  x CDATA #REQUIRED
  y CDATA #REQUIRED>
```

Some additional data may be provided in the case that the well-known subelements of `graphics` are not sufficient. The data element is meant as a preliminary means of data representation. All types of data should ultimately be converted to well-known elements. Currently this element is unused.

```
<!ELEMENT data (#PCDATA)>
<!ATTLIST data
  type CDATA #REQUIRED>
```

1.2 An example net

In this section we display an example net file as generated by the tool. The example is shortened, so that the main characteristics become more visible. In Figure 1.1, the net that is represented in this file is shown.

The `DOCTYPE` might be left out in future versions, because nets in XML-format can be parsed without a DTD.

1.3 Closing Remarks

The file format can represent both graphical information and topological information. Textual inscriptions are represented in textual form. No effort is made to represent the net entirely on a semantic level, but by using element types a tool may give certain hints what an inscription is supposed to denote. By allowing to omit the graphics part, non-graphical tools like analysers or net generators may read and write such files, too.

```
<?xml version="1.0"?>
<!DOCTYPE net SYSTEM "http://www.renew.de/xrn1.dtd">
<net id="N" type="hlnet">
  <place id="I3">
    <graphics>
      <size w="20" h="20"/>
      <offset x="48" y="36"/>
      <fillcolor><RGBcolor r="112" g="219" b="147"/></fillcolor>
      <pencolor><RGBcolor r="0" g="0" b="0"/></pencolor>
      <textcolor><RGBcolor r="0" g="0" b="0"/></textcolor>
    </graphics>
    <annotation id="I6" type="initialmarking">
      <text>1</text>
      <graphics>
        <size w="7" h="15"/>
        <textsize size="12"/>
        <offset x="0" y="0"/>
        <fillcolor><transparent/></fillcolor>
        <pencolor><transparent/></pencolor>
        <textcolor><RGBcolor r="0" g="0" b="0"/></textcolor>
      </graphics>
    </annotation>
  </place>
  <transition id="I1">
    <graphics>
      ...
    </graphics>
  </transition>
  <arc id="I2" source="I3" target="I1" type="ordinary">
    <graphics>
      <fillcolor><RGBcolor r="112" g="219" b="147"/></fillcolor>
      <pencolor><RGBcolor r="0" g="0" b="0"/></pencolor>
      <textcolor><RGBcolor r="0" g="0" b="0"/></textcolor>
      <point x="48" y="75"/>
    </graphics>
    <annotation id="I4" type="expression">
      <text>x</text>
      <graphics>
        ...
      </graphics>
    </annotation>
  </arc>
  <annotation id="A1" type="name">
    <text>demo</text>
  </annotation>
  <annotation id="I5" type="declaration">
    <text>int x;</text>
    <graphics>
      ...
    </graphics>
  </annotation>
</net>
```

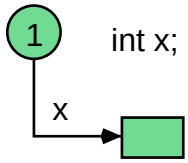


Figure 1.1: An XML file and the associated net

Bibliography

[1] Proceedings on the meeting on XML/SGML based interchange formats for Petri nets, 2000.

WWW page at `http://www.daimi.au.dk/pn2000/Interchange/`.