The International Workshop
on Discrete-Event System Design, DESDes'01,
June 27÷29, 2001; Przytok near Zielona Gora, Poland

XML APPLICATION FOR MODELLING AND SIMULATION OF CONCURRENT CONTROLLERS

Agnieszka WĘGRZYN, Piotr BUBACZ

Computer Engineering and Electronics Institute, Technical University of Zielona Góra,
ul.Podgórna 50, 65-246 Zielona Góra, POLAND, <*A.Wegrzyn, P.Bubacz*>*@iie.pz.zgora.pl*

Abstract. *The Extensible Markup Language (XML) is a subset of SGML that is completely described in this document. Such language can be used for modelling Petri nets. XML is used, because it can be exchanged between different systems. On the other hand, XML format is platform-independent, well supported, and license-free. XML is a set of rules, guidelines, and conventions, whatever you want to call them, for designing text formats for such data, in a way that produces files that are easy to generate and read. In this paper, a possibility of XML modelling and simulation of Petri net is presented.*

Key Words. *Concurrent Controllers, Petri Nets, XML, Modelling, Simulation*

1. INTRODUCTION

Petri net is a graphical and mathematical modelling tool. It can describe processing systems, which are characterized as being concurrent, asynchronous, distributed, nondeterministic [6]. In the presented method, Petri net is used for modelling of digital circuits, especially concurrent controllers.

In the paper a new way of describing of Petri net is presented. The proposed method is based on the Extensible Markup Language (XML).

XML describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879]. By construction, XML documents are conforming SGML documents [8].

The next step (after modelling) of design of digital circuit is analysis. There are different methods of verification of modelled system. One of the simplest methods is simulation. It is similar to the debugging of program execution. During simulation it is possible to check whether a circuit modelled by a Petri net behaves correctly. Therefore it is possible to recognise and remove errors in the controllers, even at early stage of design.

Several specification and design techniques based on Petri nets have been proposed [1,4,7]. They are based on software tools that help designers to develop and simulate (animate) the

system at a conceptual and abstract level. They are usually very formal and oriented towards the verification (simulation) aspects of design.

2. BACKGROUND

In this chapter basic information about Petri net and XML application is presented.

2.1. Petri nets

Petri nets are mathematical object that exists independently of any physical representation. The nets can effectively describe parallelism. The graphical form of a Petri net is used as a tool for the modelling and analysis of digital circuits, especially concurrent controllers.

A Petri net is a well-known mathematical formalism. There are different classes of Petri nets [6]. However, for the modelling of digital systems only selected classes are applicable. In this paper, coloured interpreted Petri nets are considered. Firstly, only a basic information and definitions are presented.

Petri net is an oriented bipartite graph with two subsets of nodes called the places and the transitions and the arcs joining places to transitions or transition to places. Petri net *PN* is a 4-tuple [2,3]:

$PN = (P, T; F, M_0)$,

where P – a finite set of places;
T – a finite set of transitions;
F – flow function (i.e. a finite set of arcs);
M_0 – initial marking.

On Fig. 1 an example of interpreted Petri net is presented.

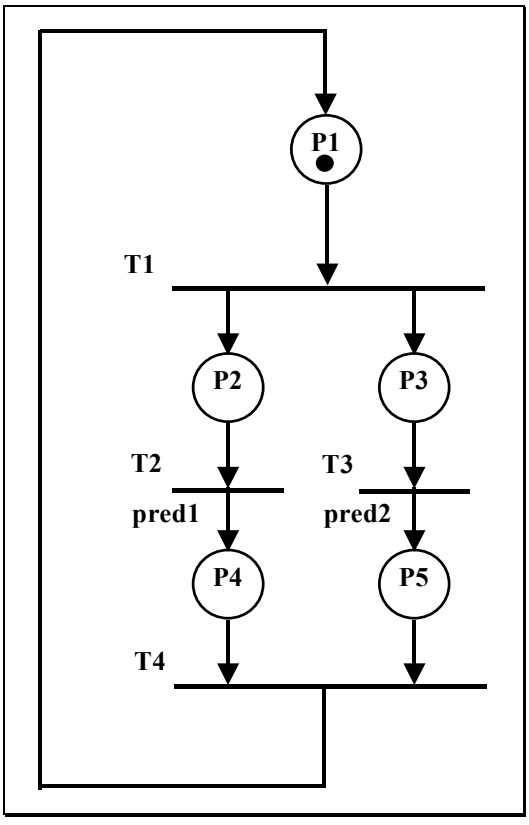

Fig. 1. An example of Petri net

2.2. XML application

The Extensible Markup Language is a set of rules for defining semantic tags that break a document into parts and identify the different parts of the document. It is a meta-markup language that defines syntax used to define other domain-specific, semantic, structured markup languages [8].

XML is a meta-markup language for designing domain-specific markup languages. Each XML-based markup language is called an XML application. This is not an application that uses XML like the Mozilla Web browser, the Gnumeric spreadsheet, or the XML Pro editor, but rather an application of XML to a specific domain such as Chemical Markup Language (CML) for chemistry or GedML for genealogy.

Each XML application has its own syntax and vocabulary. This syntax and vocabulary adheres to the fundamental rules of XML. This is much like human languages, which each have their own vocabulary and grammar, while at the same time adhering to certain fundamental rules imposed by human anatomy and the structure of the brain.

XML is an extremely flexible format for text-based data. The reason XML was chosen as the foundation for the wildly different applications discussed in this chapter is that XML provides a sensible, well-documented format that's easy to read and write. By using this format for its data, a program can offload a great quantity of detailed processing to a few standard free tools and libraries. Furthermore, it's easy for such a program to layer additional levels of syntax and semantics on top of the basic structure XML provides.

2.3. Petri Net Specification Format 3

Petri net can be present in textual form as a set of rules. Petri Net Specification Format 3 (PNSF3) is one of such textual formats. The format is based on Extensible Markup Language. PNSF3 is developed at Technical University of Zielona Gora and it based on PNSF2. By using PNSF3 an interpreted, hierarchical and coloured Petri net can be modelled.

The PNSF3 format specifies the structure of Petri net. PNSF3 does not keep information about placement of places, transitions, arcs, etc. It keeps information about names of places, transitions, number of markers, connection between places and transitions. That's why this format is simple to create and modify. PNSF3 format is easy to parse and converse to various representations. Such a way of describing creates new possibilities for example of simulation of Petri net using scalable vector graphics.

Most Petri-net research groups have their own software packages and tools to assist the drawing, analysis and simulation of various applications. They have their own Petri net format, too [5]. Now, XML based format helps to exchange data between various tools. Such approach gives possibility to verification new methods of analysis using well know and tested systems. The old formats of describing Petri net are not quite enough to apply for such verification. The goals for new XML based format are [8]:

- flexible (extendable),
- complex description,
- platform independent,
- human readable,
- easy to parse and transform.

Presented format (Fig. 2) is different from XML based format using in another Petri net tools. It differs in stored information about Petri net. In other well know systems for modelling and

analysis, XML based format keeps information about placement of elements of Petri net [5]. Preparing such format without graphical editor is very difficult. Because system in which PNSF3 is used, have no graphical editor; there cannot be store information about placement.

```
<?xml version="1.0"
encoding="ISO-8859-2" standalone="yes"?>

<pnsf3>

<clock>CLOCK</clock>

<input id="x1"> </input>
<output id="S1"> </output>
<output id="R1"> </output>

<place id="P1">
      <initmark id="m1">
            <noofmark>1</noofmark>
      </initmark>
</place>
<place id="P2" />
<place id="P3" />
<place id="P4" />
<place id="P5" />

<predicate id="pred1">x1</predicate>
<predicate id="pred2">!x1</predicate>

<transition id="T1" />

<transition id="T2">
      <condition id="c1">pred1
      </condition>
</transition>

<transition id="T3">
      <condition id="c2">pred2
      </condition>
</transition>

<transition id="T4" />
```

```
<net>
      <arc>
            <inplace>P1</inplace>
            <outplace>P2</outplace>
            <outplace>P3</outplace>
            <trans>T1</trans>
      </arc>

      <arc>
            <inplace>P2</inplace>
            <outplace>P4</outplace>
            <inpred>pred1</inpred>
            <trans>T2</trans>
      </arc>

      <arc>
            <inplace>P3</inplace>
            <outplace>P5</outplace>
            <inpred>pred2</inpred>
            <trans>T3</trans>
      </arc>

      <arc>
            <inplace>P4</inplace>
            <inplace>P5</inplace>
            <outplace>P1</outplace>
            <trans>T4</trans>
      </arc>
</net>

<MooreOutputs>
      <placeMoore>P2</placeMoore>
      <postcon>S1</postcon>
</MooreOutputs>

<MooreOutputs>
      <placeMoore>P4</placeMoore>
      <postcon>R1</postcon>
</MooreOutputs>
</pnsf3>
```

Fig. 2. PNSF3 for Petri net (Fig. 1)

3. SCALABLE VECTOR GRAPHICS

Scalable Vector Graphics (SVG) was created by the World Wide Web Consortium (W3C), the non-profit, industry-wide, open-standards consortium that created HTML and XML, among other important standards and vocabularies. Over twenty organizations, including Sun Microsystems, Adobe, Apple, IBM, and Kodak, have been involved in defining SVG [9].

SVG is a language for describing two-dimensional graphics in XML. SVG allows for three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images and text. Graphical objects can be grouped, styled, transformed and composited into previously rendered objects. A text in any XML namespace can be suitable to the application, which enhances searchability and accessibility of the SVG graphics. The feature set includes nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility [9].

SVG drawings can be dynamic and interactive. The Document Object Model (DOM) for SVG, which includes the full XML DOM, allows for straightforward and efficient vector graphics animation via scripting. A rich set of event handlers such as onmouseover and onclick can be assigned to any SVG graphical object. Because of its compatibility and leveraging of other Web standards, features like scripting can be done on SVG elements and other XML elements from different namespaces simultaneously within the same Web page [9].

3.1. SVG Features

SVG has many advantages over other image formats, and particularly over JPEG and GIF, the most common graphic formats used on the Web today. Specifically [9]:

- Plain text format - SVG files can be read and modified by a range of tools, and are usually much smaller and more compressible than comparable JPEG or GIF images.
- Scalable - Unlike bitmapped GIF and JPEG formats, SVG is a vector format, which means SVG images can be printed with high quality at any resolution, without the "staircase" effects you see when printing bitmapped images.
- Zoomable - You can zoom in on any portion of an SVG image and not see any degradation.
- Searchable and selectable text - Unlike in bitmapped images, text in SVG text is selectable and searchable. For example, you can search for specific text strings, like city names in a map.
- Scripting and animation - SVG enables dynamic and interactive graphics far more sophisticated than bitmapped or even Flash images.
- Works with Java technology - SVG complements Java technologies' high end graphics engine, the Java 2D API.
- Open standard - SVG is an open recommendation developed by a cross-industry consortium. Unlike some other graphics formats, SVG is not proprietary.
    - True XML - As an XML grammar, SVG offers all the advantages of XML:
    - Interoperability;
    - Internationalization (Unicode support);
    - Wide tool support;
    - Easy manipulation through standard APIs, such as the Document Object Model (DOM) API;
    - Easy transformation through XML Stylesheet Language Transformation (XSLT).

4. FROM PNSF3 TO SVG

The main advantage of XML is that it is very easy to transform into another format (e.g. SVG). In this chapter a way of transformation from XML based format (PNSF3) into SVG is presented.

A large majority of generally available XML based format use XSL to transformation into another format. XSL is a stylesheet language for XML. XSL specifies the styling of an XML document by using XSLT, to describe how the document is transformed into another.

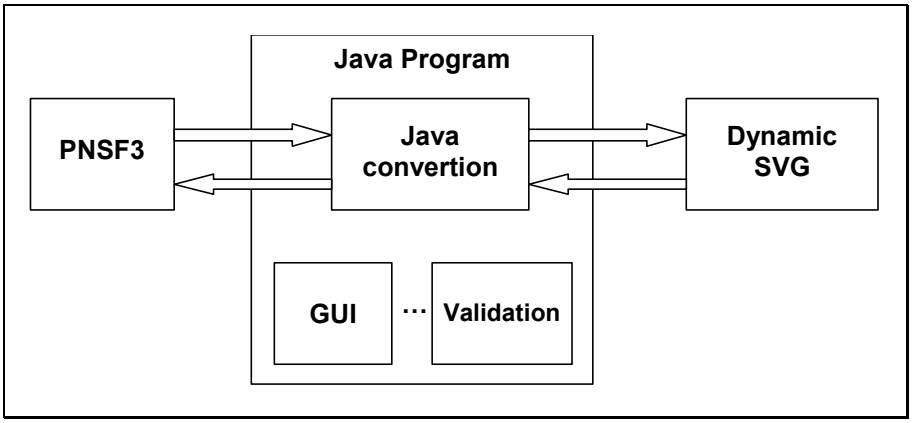

Fig. 3.Diagram of method of transformation

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN"
      "http://www.w3.org/TR/2000/CR-SVG-20001102/DTD/svg-20001102.dtd">

<svg width="500" height="500">
<g>
      <ellipse cx="231.424" cy="76.3614" rx="20.5" ry="20.5"
            style="stroke-width:1;stroke-opacity:1;stroke:rgb(0,0,0);
            fill-opacity:0;fill:rgb(0,0,0);opacity:1"/>

            <text x="219.032" y="84.4564" style="font-family:Arial;font-size:24;
                  stroke-width:1;stroke-opacity:1;stroke:rgb(0,0,0);
                  fill-opacity:1;fill:rgb(0,0,0);opacity:1">P1</text>

            <path d="M232.125 23.0995 L232.125 56.0114"
                  style="stroke-miterlimit:4;stroke-linejoin:miter;
                  stroke-linecap:round;stroke-width:1;stroke-opacity:1;
                  troke:rgb(0,0,0);fill-opacity:1;fill:rgb(0,0,0);
                  opacity:1"/>

            <path d="M230.315 130.331 L225.457 122.902"
                   style="stroke-miterlimit:4;stroke-linejoin:miter;
                   stroke-linecap:round;stroke-width:1;stroke-opacity:1;
                   stroke:rgb(0,0,0);fill-opacity:1;fill:rgb(0,0,0);
                  opacity:1"/>

            <path d="M231.086 130.293 L235.944 122.864"
                  style="stroke-miterlimit:4;stroke-linejoin:miter;
                  stroke-linecap:round;stroke-width:1;stroke-opacity:1;
                  stroke:rgb(0,0,0);fill-opacity:1;fill:rgb(0,0,0);
                  opacity:1"/>

      . . .
</g>
      <ellipse cx="230.5" cy="87.5" rx="4.5" ry="4.5" style="stroke-width:0;
            stroke-opacity:1;stroke:rgb(255,255,255);
            fill-opacity:1;fill:rgb(255,255,255);;opacity:1">

            <animateColor attributeName="fill" attributeType="CSS"
                        values="black;white;white;white;white;white"
                        dur="6s" repeatCount="indefinite"/>
      </ellipse>
      . . .
```

Fig. 4. A part of SVG file

In section 2.3 it is mentioned, that PNSF3 do not keep information about placement. Because of this fact, it is difficult to directly make XSL file for transformation into SVG. For such

case, a special program should be created to prepare XSL file. SVG file can be made in the other way, too. On Fig. 3 diagram of method of transformation from PNSF3 into SVG file, without preparing a XSL file, is presented. For the conversion a Java application is used. The program generates a dynamic SVG file. The presented part of file (Fig. 4) is prepared for PNSF3 (Fig. 2). SVG file keeps information about places, transitions, predicates, markings and placement of these elements. Using option of animation of dynamic SVG, it is possible to simulate a Petri net. At present stage, simulation of Petri net without interpretation is made. The next step of researches will be studying possibilities of simulation of interpreted Petri net.

5. CONCLUSIONS

In the paper a new format (PNSF3) based on XML for describing Petri net, is presented. The benefit of this format is that it allows easy changes and transformations to other formats. PNSF3 is described in XML syntax because XML parsers are easily available and XML is becoming an international standard. In the presented format, placement and set of rules are separated. It is both, advantage and disadvantage, because the format is very easy for preparing and change, but a bit difficult for converting into a graphical format. This problem is solved by application, which generate graphical format. Using of XML for describing Petri net makes it possible to simulate Petri net, by using dynamic SVG.

REFERENCES

[1] M.Adamski, M.Wegrzyn, "Hierarchically Structured Coloured Petri Net Specification and Validation of Concurrent Controllers", *39th International Scientific Colloquium IWK'94*, 1994, Ilmenau, Germany, Band 1, pp.517-522

[2] E.Best, C.Fernandez, "Notations and terminology on Petri net theory", *Petri Net Newsletter*, 1986

[3] R.David, H.Alla, *Petri Nets & Grafcet. Tools for modelling discrete event systems*, Prentice Hall, New York, 1992

[4] L.Ferrarini, "An Incremental Approach to Logic Controller Design with Petri Nets", *IEEE Transactions on System, Man, and Cybernetics*, Vol.22, No.3, May/June 1992, pp.461-474

[5] R.B.Lyngsø, T.Mailund, "Textual Interchange Format for High-Level Petri Nets", *Workshop on Practical Use of Coloured Petri Nets and Design*, June 1998, Aarhus University, pp.47-64

[6] T.Murata, "Petri Nets: Properties, Analysis and Applications", *Proceedings of the IEEE*, Vol.77, No.4, April 1989, pp.541-580

[7] J.L.Peterson, *Petri Net Theory and The Modeling of Systems*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1981

[8] Extensible Markup Language (XML) 1.0. W3C Recommendation, *http://www.w3.org*

[9] Scalable Vector Graphics (SVG) 1.0 W3C Candidate Recommendation, *http://www.w3.org*