Towards Petri Net WWW Community: Starting Points

Dragan Gaševic

Military Academy, Department of Computer Science
Ratka Resanovica 1, 11133 Belgrade, Serbia and Montenegro
gasevic@yahoo.com

Vladan Devedžic

FON – School of Business Administration, University of Belgrade
Jove Ilica 154, POB 52, 11000 Belgrade, Yugoslavia
devedzic@galeb.etf.bg.ac.yu

Abstract

This paper presents basic guidelines for building Petri net ontology, which should enable using Petri nets on the Semantic Web and help establish a true Web community of Petri net users and applications. Available Petri net model descriptions give format definitions. Their main objective is to provide the syntax for sharing Petri net models between different applications. Some of these applications are software-specific and provide either text-based format definitions (e.g., DaNAMiCS) or XML-based format definitions (e.g., Renew). Another kind of such applications proposes a universal syntax definition for model sharing, for example Abstract Petri net notation (BNF definition) and Petri Net Markup Language (XML definition). The main drawback of these solutions is their emphasis on syntax per se, ignoring the fact that syntax represents means for sharing semantics of Petri net models. Assuming that syntax is just a medium for model transportation, this paper considers the basic concepts and their relationships in existing Petri net formats. The ontology guidelines we present are based on an analysis of such formats and are represented using UML.

Keywords: Petri nets, Petri nets formats, XML, PNML, ontology, UML.

1. Introduction

Petri nets are formal mathematical and graphical tool for modeling, simulation and analysis of processes specially distributed [1]. There are many software solutions as their support (Design/CPN, Petri Net Kernel, DaNAMiCS, Renew) [2]. Each implementation has different capabilities of Petri net using. The paper [3] emphasises the fact that it would be very useful if Petri net researchers would share their Petri net model descriptions.

There are two possible solutions for using different softwares for the same model observation [4]. The first solution implies that a software solution implements as many parsers as the number of softwares to perform the exchange with. The second way is general syntax formulation for Petri net models description by which different softwares solutions would be used describe their models. The syntax definition should contain a set of elements (tags) which should be predefined by introducing the convention. This convention would define the meaning of each element.

However, Petri net using on the Semantic Web or by Web services does not imply just Petri net model exchange considering only syntax for document writing. This also implies consensus of concept meanings which are exchanged by syntax, whereas the syntax is only a medium for semantic

exchange. Therefore, Petri net ontology definition [5] is needed for semantic description of Petri net concepts and their relationships.

The basic assumption used in this paper is that there are a great number of formats and languages for Petri net model markup. They are generally accepted by the Petri net community and used in current Petri net software implementations. For that reason, analysis of these formats and languages is suggested with the aim to identify common concepts described in them.

Two universal syntaxes for model exchange are taken for the subject analyses: Abstract Petri Net Notation (APNN) and Petri Net Markup Language (PNML). The APNN is analysed (although it is not used by any software) because it is the first attempt to define Petri net universal syntax which includes different Petri net dialects. The PNML is analysed in the light of software solutions which use it – Petri Net Kernel and P3. Beside other Petri net software specific formats, the following formats are analysed: DaNAMiCS (text format) and Renew (XML format).

The next section is about the descriptions of the Abstract Petri Net Notation, the DaNAMiCS and the analysis of its format for model exchange, and Renew software solution and its XML format. The analysis of PNML usage is given in the third section. The fourth section contains the comparative review of analysed formats for Petri net model description as well as basic guidelines for Petri net ontology in the form of UML model. The last section contains final conclusions.

2. Review of analized formats: Abstract Petri Nets Notation, DaNAMiCS forma, Renew XML format

Abstract Petri Net Notation is presented in the paper [6]. It can describe different Petri net dialects. To increase the readability of this notation, the key words are similar to LATEX commands. This notation should satisfy the following requests: net descriptions could be easily exchanged in electronic form, *extensibility* (it can be used by different Petri net dialects. Simple Petri net dialects can be extended in order to describe high-level dialects), *modularity and hierarchy* (the Petri net description in the file should be such that it can be reused), *readability* (text notation should be such that it can simply be transformed into the human-readable format as well as suitable for printing). This notation does not save information of Petri net graphical elements (place position, transition, place name, etc.). Abstract notation for each Petri net class is defined in Backus Naur Form (BNF) Notation.

For model record DaNAMiCS [7] software tool uses a file format with the *bim* extension. Format with the *bim* extension has many internal marks whose documentation is not available for the authors of this paper. The second file format which DaNAMiCS uses has the *wam* extension and these files are used for model import (menu *File*, option *Import net*). The authors are not familiar if this format has the available documentation of its description. However, this format is textual with an evident structure that corresponds to certain Petri net object. This format has been analysed and compared with the models obtained by importing files of this format into the DaNAMiCS and the meaning of every format element has been obtained as well.

The Petri net basic concepts are represented by the following labels: `Places`, `ImmedTrans` (and `TimedTrans` – since DaNAMiCS supports Stochastic Petri net and timed transitions) and `Edges`, respectively. These labels also represent blocks in which the records of each element according to type are. Within the record of a certain element there are its basic attributes – drawing position (graphical information), initial marking, arc marking, place and transition name. `PiPs` and `TiPs` labels are for page (Petri net structuring mechanism) representation.

In order to overcome the problem of model exchange with other Petri net software solutions, Renew uses XML. The XML document model description is defined by *Document Type Definition* (DTD) [8, 9]. The assumptions, which are included in the formulation of this DTD, are the same as the PNML assumptions since there are the same elements for the description of: net (XML tag `net`), place (`place`), transition (`transition`) and arc (`arc`). These elements in these two formats also have similar content model. Each element in the Renew's XML format can have graphical information and arbitrary number of annotations. The graphics information is a set of information comprising the following elements: size (`size`), text size (`textsize`), relative position (`offset`), object fill color (`fillcolor`), color of object outline (`pencolor`), text color (`textcolor`), absolute position (`point`), arbitrary data (`data`).

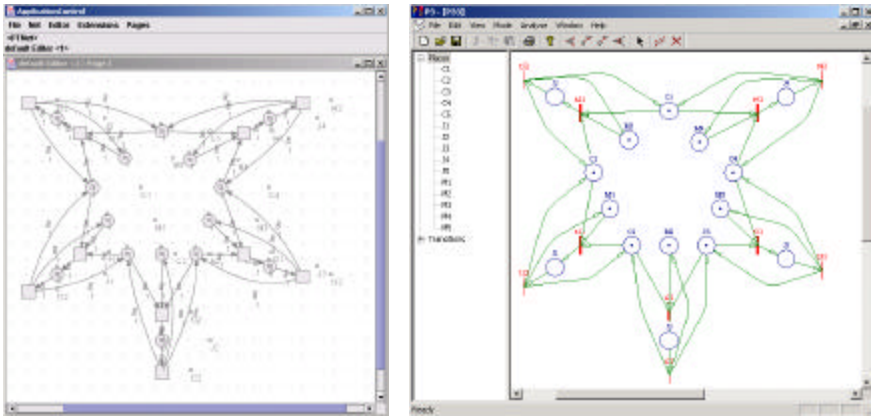
3. Petri Net Markup Language – proposal for standard of Petri net markup

The *Petri Net Markup* Language (PNML) [10] is defined by the eXtensible Markup Language (XML). The PNML is not *de jure* standard, but it is generally accepted by the Petri net society being constantly in progress [11]. The proposed basic language structure for the Petri nets universal markup consists of two parts: a general one (which is independent of specific Petri net dialects) and a specific one. The general part is called Petri Net Markup Language (PNML) and the specific one is Petri Net Type Definition (PNTD). The PNML definition is given by using Tree Regular Expressions for the XML (TREX) and the XML Schema [12].

Two softwares have been analysed in order to illustrate the current state of PNML appliance. The first software solution is Petri Net Kernel (PNK) in whose context the PNML has been developed. The second software is P3, which has been developed by authors in order to explore the applicable capability of the existing PNML definition. Further in this paper a short review of these two projects has been given.

The Petri net software solution PNK has been downloaded from [13], it is available for free and implemented in Java (figure 1a). The PNK is an infrastructure for building Petri net tools [14] – it is not just a Petri net tool. It is rather an Application Programming Interface (API) for building Petri net tools. It is not limited to one Petri net dialect and can be used for each Petri net dialect, which have their own specifics. A PNK based tool consists of two kinds of modules : the application functions and the PNK modules. The application functions are implemented by a PNK user, whereas the PNK modules are included in the PNK. Of course, there are several application functions (such as the graphical editor) within the very PNK distribution. The implemented Petri net dialects by the public available version of PNK are the following: P/T net, High-level Petri net, Time Petri net, Timed Petri net, bag net, echo Net, Black Token net. The PNK also implements: graphs and ghs graphs. The PNML is used by PNK for model exchange with other software solutions.

P3 software solution [4] is implemented in MS Visual C++ IDE and intended for using on Windows operating system platform (figure 1b). It supports Petri net dialects: P/T net, Upgraded Petri net. Analysis tools, which are implemented by P3, are the following: reachability tree, matrix equations, matrix invariants, fireability tree, non-cyclic graph of firing. Two kinds of execution of a simulation are implemented: individual execution of fireable transitions and parallel execution of fireable transitions. P3 uses the PNML for model exchange with other software solutions. The PNML extension for markup of Upgraded Petri net is made within this software. P3 uses the PNML definition given in the form XML Schema [12].



a) b)

Figure 1. Softwares that use PNML: a) Petri Net Kernel, b) P3

The unmodified PNML definition in the form of the XML Schema document, which is downloaded from [12], is used by P3. However, the PNK uses a little extended definition of the PNML, compared to the P/T net description. The difference is in the content model of element for place tag (`place`). In

content model of this element there is the element `marking` at the first place – its content model is the same with the content model of initial marking (`initialMarking`). This element is for description of current marking of places. Elements for position description: absolute (`position`) and relative (`offset`) have an attribute `page`, which is not given in the PNML definition. Position of a model object in a graphical representation on a page is described by this attribute. Difference related with arcs is that the PNK does not allow arcs to have more than one intermediate point (which is required), in contrast to P3 and the PNML definition which allow using more than one intermediate point. It is also possible that there is not a single intermediate points.

4. Towards Petri Net WWW Community

Petri net formats and markup languages, which are analysed in previous sections, put syntax for Petri net description in the first place. However, in order to obtain these grammar definitions, it was necessary to make certain conceptual assumptions as their base. This mostly refers to the universal exchange syntax: the APNN and PNML whereas internal formats of analysed softwares deal more with syntax. Regardless of these facts, it is possible to distinguish few common concepts, which describe the domain of Petri nets. The review of common concepts is made in table 1. Attributes (normal face font) of particular concepts as well as their related elements (bold face font) for each format are distinguished. This also constitutes basic guidelines for building Petri net ontology.

Table 1. Review of Petri net concepts, attributes and content

Concept	APNN	DaNAMiCS	Renew	PNML
Net	Identifier, **place**, **transition**, **arc**	-	Identifier, type, **place**, **transition**, **arc**, **annotation**	Identifier, type, **place**, **transition**, **arc**, **page**, **reference place and transition**
Place	Identifier, **name**, **initialmarking**, **capacity**	Name, initial marking, marking, graphical information	Identifier, tip, **graphical information**, **annotation**	Identifier, **name**, **initial marking**, **graphical information**
Transition	Identifier, **name**	Kind (immediate, time), name, graphical information, possibility, time, delay	Identifier, tip, **graphical information**, **annotation**	Identifier, **graphical information**, **name**, **toolspecific**
Arc	Identifier, source, target, multiplicity	Source, target, multiplicity, graphical information	Identifier, source, target, type, **graphical information**, **annotation**	Identifier, source, target, **graphical information**, **multiplicity**
Graphical information	-	Position	**Position**, **size**, **text size**, **color**, …	**Absolute position**, **relative position**
Initial marking	Init	Contained by place tag	Annotation: type, identifier, **text**, **graphical information**	Value, **graphical information**
Name	Value	Contained by place tag	Annotation: type, identifier, text, **graphical information**	Value, **graphical information**

The authors suggest the use of the PNML as a base for Petri net ontology. The reason is that it represents a language for Petri net markup, which included the following: the fact that there are many Petri net dialects, possibility of the extension of some formats of Petri net dialects with additional features by using object oriented principles, and the fact that it should be a part of Petri net standard. A special advantage of using the PNML is the fact that it contains all concepts founded in each format, which have already been analysed in this paper.

The proposed approach for building ontology is based on the idea formulated in [15], who says that syntax is a medium for semantics. In this case syntax is a guideline for the ontology. In order to obtain the full notion of ontology, the authors suggest the use of the UML [16]. The use of the UML is suitable because it is generally accepted and standardised for analysis and modeling in software

engineering. The UML does not have a graphic notation only, but also profiles, global modules, and extension mechanisms [17]. You can directly obtain the Petri net ontology by using the UML, as well as ontology mapping into some target format – based on DTD, XML Schema, RDF, etc. According to this recommendation the UML model of PNML XML Schema definition has been obtained by reverse engineering procedure. For that purposes the XSL (Extensible Stylesheet Language) transformation, which transforms a XML Schema definition document into XMI (XML Metadata Interchange) format, has been developed. An example of a class diagram, which was obtained by this transformation, is given in figure 2. XMI is OMG's standard format for Meta Object Facility compilant languages, like UML is. That diagram shows basic Petri net object of the PNML taxonomy. They are modelled by the UML profile for modeling XML Schema, which is given in [18]. The UML model is represented using XMI, so it is possible to use different UML tools implementing XMI support (for example, Rational Rose, ArgoUML, Poseidon for UML). In this paper, the Rational Rose UML tool has been used in order to gain the Petri net ontology and all modifications purposed in the following text had been done using it.

The reverse XML Schema represented by the extended UML model is a starting point for the Petri net ontology definition. In the next ontology development stage, it is necessary to do some changes on the UML model in order to equalize the UML model terminology to general-accepted Petri net concepts, as well as to delete construction used for XML Schema modeling. From the obtained model one can see that class names did not match the concept names general accepted by Petri net community, thus it is necessarily to rename them for precise ontology definition, for example *typePlace* to *Place* or *typeArc* to *Arc*. Also, the obtained UML model shows that the PNML, as the syntax for Petri net model sharing, contains some syntax constructions that do not have some special semantic meaning for Petri net models interpretation, but are representing just syntax features. An example of these constructions is an XSDcomplexType stereotype *pnml* that represents syntax tag for PNML document, while it is not a real Petri net concept. This construction group contains some XML Schema specific stereotypes as well, like XSDchoice, XSDsequence, and XSDany. When these constructions have no meaning to the Petri net ontology, they have to be removed from the UML model. In order to avoid losing information about the associations for which these UML elements are used, it is necessary to create the associations using the elements that are related through the deleted ones. The changed model in the class diagram form is given in the figure 3 (Rational Rose model).

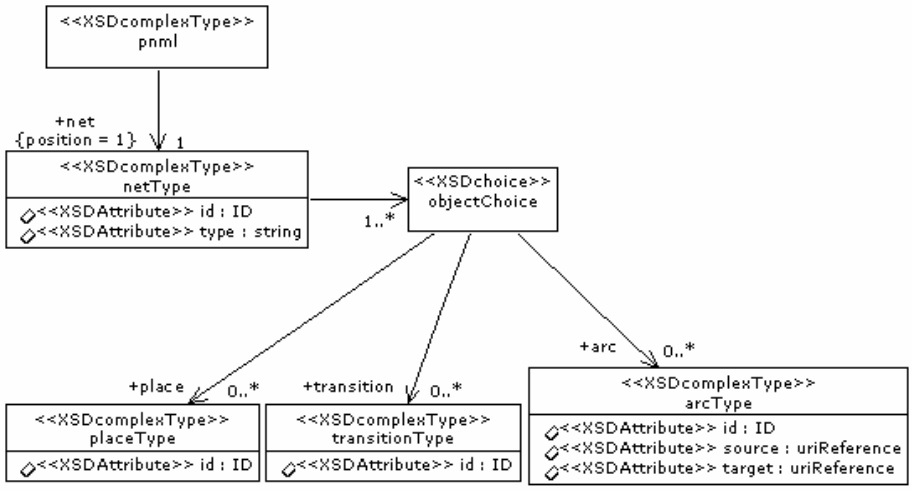


Figure 2. PNML Class Diagram – PNML document content model description

However, neither Rational Rose nor the UML itself are tailored specifically for ontology development. In order to enable more precise Petri net definition than the model obtained in the first stage of the model modification, it is necessarily to enable its usage by some ontology development tool, like Protégé 2000 is [19]. One of the possible solutions is to represent the described UML model using the RDF Schema language, which intends to represent ontology on the WWW. The RDF Schema language is supported by Protégé. The XPetal (which is implemented in Java and transforms a mdl Rose file to a RDF Schema) software tool [20] has been used for transformation from the UML

model to the adequate RDF Schema. The UML model represented by Rational Rose as well as it's equivalent represented by Protégé are shown in the figure 3.

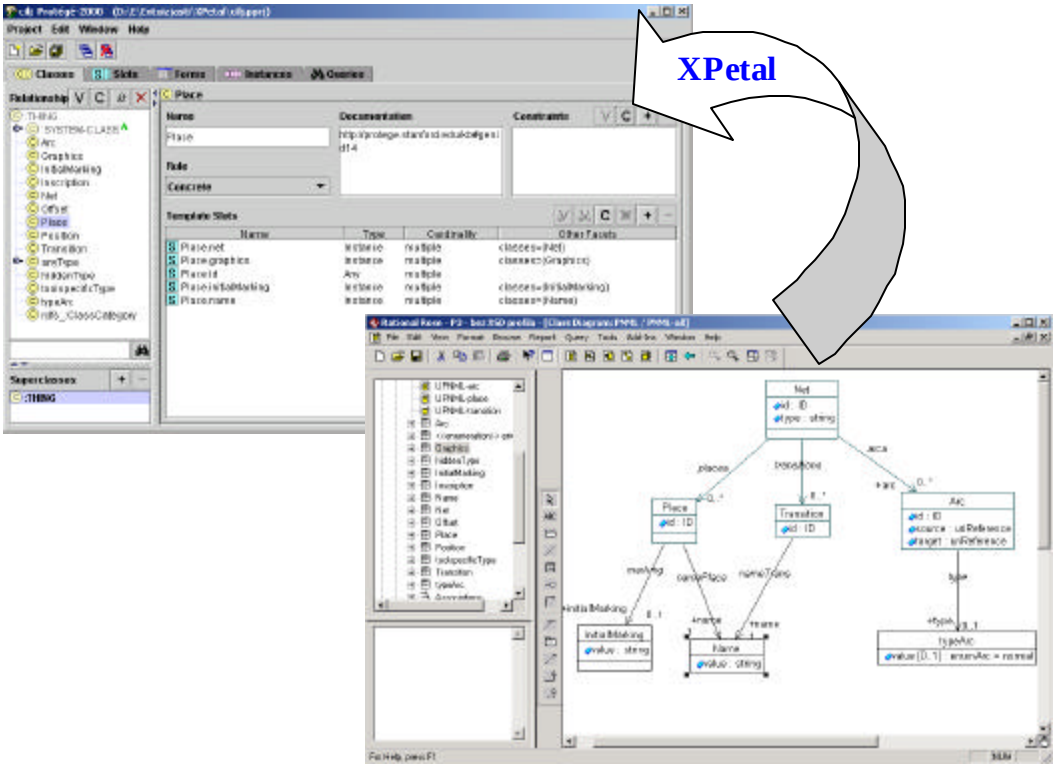


Figure 3. Petri net ontology exchange between Rational Rose and Protégé using RDF Schema and XPetal

The next stages of Petri net ontology development consist of inserting additional elements, that are increasing the abstraction level of some concepts (for example, Place and Transition are Node), as well as appending concepts founded in other Petri nets sharing languages. This also means adding different constraints defined by Petri net definition itself (arc between two different nodes). Further, a distinct aspect of Petri net ontology are Petri nets mechanisms structuring support (for example, pages, modules). Also, when an UML model can not represent all constructions for ontology modeling that Protégé supports, as well as other languages for ontology representation (RDF Schema, DAML, OIL, DAML+OIL, OWL), then it is proposed to use UML profile that will extend the UML metamodel with concepts needed for ontology modeling [17].

5. Conclusion

The review of Petri net formats (DaNAMiCS, Renew) and markup languages (Abstract Petri Net Notation and Petri Net Markup Language) has been made in this paper. By comparative analysis of elements which are contained in these formats, certain concepts, their attributes and relationships between them have been obtained. These concepts make the starting point for Petri net ontology development, as a major vehicle towards building WWW community of Petri net users, applications, and services. According to the fact that the PNML is a language in constant progress which is to become a standard language for Petri net markup, the authors propose its use as an ontology base. It contains all concepts as well as other analysed languages, and it could be extended.

The main contribution of this paper is the guidelines given for Petri net ontology. These guidelines are obtained by analysing the current formats for Petri net model exchange. The originality of the proposed solution is the fact that the ontology building is based on the existing Petri net syntaxes. Furthermore, by defining Petri net ontology it is possible to use them on the Semantic Web and by Web services. This means that the description of some Petri nets can be found in a document which is

not necessarily for Petri net softwares use. For instance, those documents can contain natural-language text described in the HTML, provided that they should contain metadata of Petri net concepts as well.

The main tendency of the future work means complete defining of the Petri net ontology. This also means translating the PNML's UML model into the UML profile for development ontologies, as well as additional consideration of relationships between concepts of Petri net ontology, because this paper analysed relationships of the has a type (having a property) only, and not relationships of the is a type. Special aspect would be consideration and adding mechanisms of structuring to the Petri net ontology. Further, a way for representation should be analysed: the relationships between different Petri net dialects, and data, for example, about model creators, the location of the theoretical definitions of applied formalisms on some model, addresses of the Web services that could provide these information or which could be used for analysis of Petri nets.

References

[1] Peterson, J. L., *Petri net theory and the modeling of systems*, Prentice Hall, Englewood Cliffs, New Jersey, USA, 1981.

[2] Petri Nets Tools Database, http://www.daimi.au.dk/Petri-Nets/tools/db.html, last visit December 2002.

[3] Berthelot, G. et al, "A syntax for the description of Petri Nets", *Petri Net Newsletter*, No. 29, 1988, pp. 4-15.

[4] Gasevic, D., "Upgraded Petri Nets Universal Markup", *MSc work*, School of Electrical Engineering, Belgrade, Serbia and Montenegro, 2002.

[5] Gruber, T., "A translation approach to portable ontology specifications", *Knowledge Acquisition*, Vol.5, No.2, 1993, pp. 199-220.

[6] Bause, F. et al, "Abstract Petri Net Notation ", *Petri Net Newsletter*, No. 49, 1995, pp. 9-27.

[7] DaNAMiCS home page, http://www.cs.uct.ac.za/-Research/DNA/DaNAMiCS/, last visit September 2002.

[8] Renew home page, http://www.renew.de/, downloaded in February 2002.

[9] Kummer, O., Wienberg, F., "The XML File Format of Renew", *Meeting report from 21st ICATPN*, Århus, Denmark, 2000.

[10] Weber, M., Kindler, E., "The Petri Net Markup Language", to appear in *Petri Net Technology for Communication Based Systems*, LNCS 2472, Springer-Verlag, Berlin-NewYork, 2003.

[11] Rémi, B. et al, Meeting on XML/SGML based Interchange Formats for Petri Nets, *Meeting report from 21st ICATPN*, Århus, Denmark, 2000.

[12] PNML home page, http://www.informatik.hu-berlin.de/top/pnml/, last visit December 2002.

[13] Petri Net Kernel home page, http://www.informatik.hu-berlin.de/top/pnk/download.html, downloaded in December 2003.

[14] Kindler, E., Weber, M., 2001. "The Petri Net Kernel - An Infrastructure for Building Petri Net Tools", *Software Tools for Technology Transfer (STTT)*, Vol.3, No.4, , Springer-Verlag, Berlin, pp. 486-497.

[15] Robie, J. et al, "The Syntactic Web Syntax and Semantics on the Web", *Markup Languages: Theory and Practice*, MIT Press, Vol.3, No.4, 2001, pp. 411-440.

[16] Cranefield, S., "Networked Knowledge Representation and Exchange using UML and RDF", *Journal of Digital information*, Vol. 1 No. 8, http://jodi.ecs.soton.ac.uk, 2001.

[17] Baclawski, K. et al, "Extending the Unified Modeling Language for ontology development", *Journal Software and Systems Modeling (SoSyM)*, Vol.1, No.2, Springer-Verlag, Berlin-NewYork, 2002, pp. 142-156.

[18] Carlson, D., *Modeling XML Applications whit UML: Practical E-Business Applications*, Addison-Wesley, Boston, USA, 2001.

[19] Noy, N. F. et al, "Creating Semantic Web Contents with Protégé-2000", *IEEE Intelligent Systems*, Vol.16, No.2, 2001, pp. 60-71.

[20] XPetal home page, http://www.langdale.com.au/styler/-xpetal/, downloaded in January 2003.