Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

REPRESENTATION AND SIMULATION OF STOCHASTIC PETRINET MODELS USING XPNML

Hyunsoo Lee
Bikram Sharda
Amarnath Banerjee

Department of Industrial and Systems Engineering
3131 TAMUS
Texas A&M University
College Station, TX 77843, U.S.A.

ABSTRACT

The paper presents an extensible Petri Net Markup Language (xPNML), which is an extended version of PNML. The xPNML format overcomes limitations associated with PNML structure for simulation and analysis of stochastic Petri nets. In addition, the proposed format supports modular concepts for creating specific models (for example, a module for machine operation including working, failed and repair states). These modules can be used as templates and exchanged among different net models. The paper provides the ontology of xPNML model and overview of its JAVA based implementation for simulation and analysis of stochastic Petri nets.

1 INTRODUCTION

Petri nets provide a graphical and mathematical framework for understanding systems that are characterized by distributed, parallel, concurrent and non deterministic/stochastic attributes (Murata 1989). Over the years, different variants of Petri nets (such as stochastic Petri nets, colored Petri nets (Jensen 1997) have been developed, and applied in areas such as distributed systems, manufacturing system modeling and control.

Today, there are more than 70 different tools available for modeling and analysis of different types of Petri nets. One of the major problems with these software is lack of a common medium to integrate the salient features of each software. For example, there is no standardized file format, by which a user can create a model in one tool, analyze/evaluate its properties in another tool. In addition, with the development of such a standardized medium, the information can be shared with different cross functional groups at different geographical locations, who can work on specific issues in Petri nets modeling (Billington et al. 2003, Weber 2002).

Another problem associated with Petri net modeling is the state space explosion problem and lack of modular concepts. Petri net models become very complex even for simpler systems, and also become difficult to comprehend as number of states grow. To overcome this problem, several new variants of Petri nets (Chung-Hsien 2005, Jensen 1997) have been suggested. One of the simpler alternative to this problem can be use of modular concepts as used in discrete event simulation tools such as ARENA <http://www.arenasimulation.com>. Modular concepts can help a great deal in creating abstract models for understanding the system while each module detail can provide information about individual subsystems in the model.

To address some of these issues, an XML based Petri Net Markup Language (PNML) <http://www2.informatik.hu-berlin.de/top/pnml/about.html> was developed as a de-facto standard for modeling Petri net models. This provides a standardized medium for creating Petri net models which can facilitate information sharing among different software tools. Despite being a good interchange methodology for Petri net definition, PNML has certain limitations.

PNML is centered around Petri net model creation in a standardized format that can be used by other software tools. However, it does not provide any information about model simulation and analysis. In addition, the PNML schema does not provide a modular method for creation of Petri net models.

To overcome these limitations, we propose an extensible Petri Net Markup Language (xPNML). xPNML builds upon the concepts of PNML, and adds a schema for simulation and analysis. In addition, xPNML provides a modular concept which aids in developing simpler and easier to comprehend Petri net models. The proposed schema is applied for stochastic Petri nets and is tested on a JAVA based Petri net simulator.

In this paper, we describe the ontology of xPNML and demonstrate how simulation and analysis information can be represented and transferred. In addition, the paper

shows how the proposed approach can be used for stochastic Petri net models.

2 BACKGROUND AND LITERATURE REVIEW

2.1 Petri nets

Petri net is a directed bipartite graph developed by Carl A. Petri in 1962. Since then, several new extensions such as colored Petri nets, stochastic Petri net, Distributed Agent Oriented Petri nets have been developed and applied in multitude of areas ranging from distributed systems, computer architectures, neural networks and manufacturing systems modeling. Readers are suggested to refer to (Murata 1989, Zhou 1999, Al-Jaar 1990, Baccelli F. 1994) for more details on Petri nets and their applications.

Figure 1 shows a FMS cell which consists of 2 material processors (MP), 1 material hander (MT) and 2 Material transporters (MT). The control sequence in this cell can be represented by Petri net model shown in figure 2. The Petri net model can represent precedence relationship between entities, and represent structural interaction of unpredictable, concurrent and asynchronous events. The Petri net model can then be used to identify bottlenecks and conflicting situations in FMS cell.

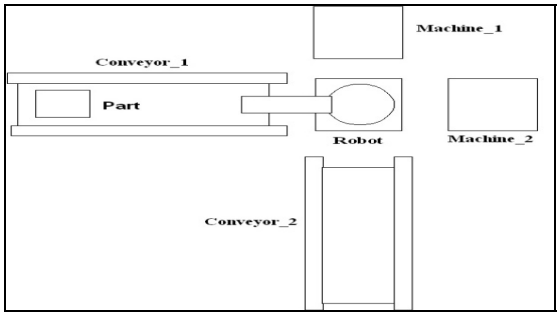

Figure 1: FMS cell.

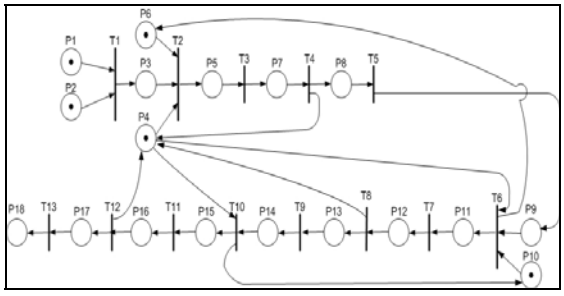

Figure 2: Petri net model for Figure 1.

In stochastic Petri nets (SPN), the time taken for transition/place firing follows a probability distribution. For example, in figure 2, the firing time for transition T9 represents time for processing the part on Machine_2, and this time can be represented stochastically by a distribution, such as exponential or normal with appropriate parameter(s).

2.2 Petri net Tools

Currently there are more than 70 stand alone software packages for modeling, simulation and analysis of different types of Petri nets. Some of these are proprietary software packages and some are open source. The current trend in a majority of these software is to use JAVA <http://java.sun.com>. This is because JAVA has several open APIs for mathematics and graphics in 2D/3D. As a result, most open source APIs of Petri net are implemented in JAVA. Readers can refer to <http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html> for a more detailed list of the tools available, and their strengths and weaknesses. Some of these tools provide good graphical interface for creating the models while some provide good interface for analysis. However, there is no single tool which is comprehensive enough to provide a single modeling, analysis and simulation platform. In addition, each of these tools have their own format for storing and retrieving models, which makes it very difficult to interchange data and information between these software.

PNML was developed to address some of these issues. It provides a standardized interchange format which can be used by different software. This helps in incorporating the salient features of each of these software. The following section describes the ontology of PNML.

2.3 Petri Net Markup Language (PNML) ontology

Petri Net Markup language (PNML) is an XML based interchangeable format for Petri Nets (Billington et al., 2003). This was originally developed as a file format for Petri Net Kernel <http://www2.informatik.hu-berlin.de/top/pnk/index.html>. PNML allows flexibility to integrate different types of Petri nets and openness to future Petri net extensions. The schema of PNML is defined in Petri Net Type Definition (PNTD) and Conventions Document. Using PNTD and Conventions Document, PNML can be extended and represented in various forms.

Figure 3 shows the overview of different parts of PNML. The Meta Model represents the basic structure of the Petri net. The Type Definition allows definition of new Petri net types and Feature Definition allows definition of new features in Petri nets. These three parts are fixed and do not change. The Conventions Document is an evolving document which contains the definition of standard features of Petri nets, defined according to Feature Definition interface. New features can be added to Conventions Document and Petri net Type Definition and are maintained via a common web site

<www.informatik.hu-berlin.de/top/pnml/>. Currently, PT-Net and HLPNG model are being fixed as a standard document.

Figure 4 shows the detail of meta model of PNML. *PetriNetDoc* meets the requirements of a PNML core model. The *PetriNetDoc* can contain multiple Petri nets (*PetriNet*) which are identified by a unique *id* and *type*. A Petri net model can have multiple top level *Pages*, which contain objects. The objects of a Petri net model are *nodes* and *arcs*, which are identified by a unique *id.* The nodes of a Petri net model are places (*PlaceNode*) and transitions (*TransitionNode*). In addition, the objects have *labels,* which are further classified in the form of *attributes* and *annotations*. For constructing high level Petri net models, PNML core model specifies *Pages* and *Reference* nodes. The PNML model file also contains tool specific information which is unique to each tool that creates a PNML file.

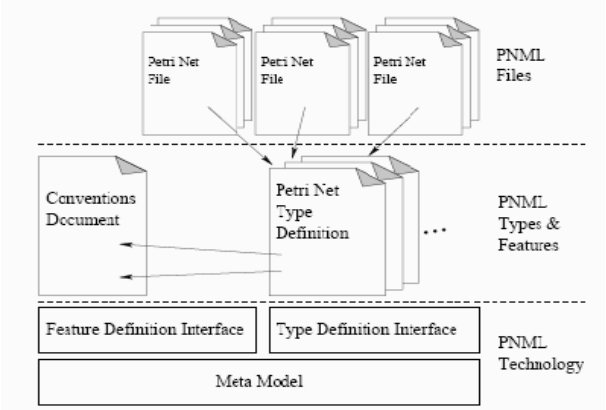

Figure 3: Overview of PNML parts (Billington et al. 2003)

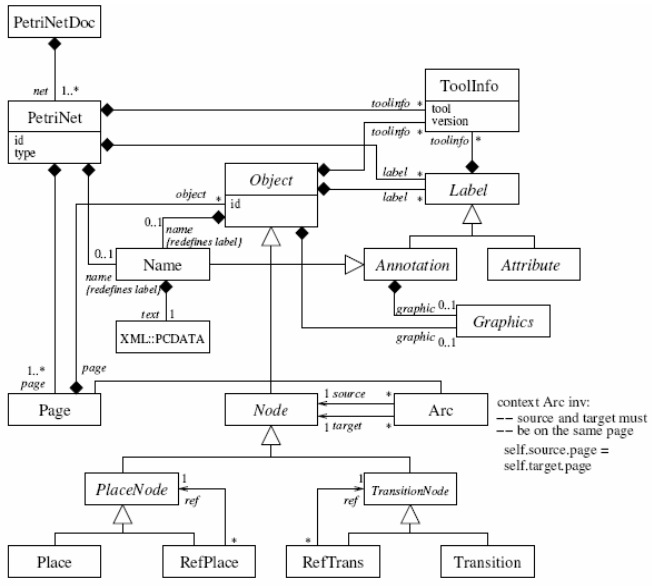

Figure 4: PNML core model (Kindler et al. 2006)

The current PNML structure does not provide any information about simulation and analysis of the Petri net model. In addition, the information contained in object annotation (such as initial marking, timing) is represented in a text format, which makes it difficult to use for simulation and analysis purposes. This information is very important for analysis of Petri net models. In addition, this information should be easily searchable in a PNML file. These features can be enabled if the schema is described in the meta model as common entity associated with each object.

3 REPRESENTATION OF PETRI NET MODEL USING XPNML

3.1 Considerations for simulation and analysis

As discussed in section 2, the information in PNML is centered around how to build Petri net models. This schema does not provide important information that is required for simulation and analysis of Petri net models, which limits the potential use of PNML. The Petri net model file should not only provide information about Petri net structure but also provide information about governing parameters which are used for analysis and simulation. This information is critical to compare alternative models as the results will be different despite same Petri net structure. Providing simulation parameters gives a standard for comparing alternative models.

xPNML focuses on how to represent Petri net model as well as incorporate information for simulation and analysis. Generally this information can be classified into simulation environment, simulation attributes and analysis parameters. Table 1 describes this information in detail.

Table 1: Information for simulation and analysis

Classification	Information requirements
Simulation environment	- Number of replications - Replication length - Warm-up period - Time units
Simulation attributes	- stochastic distribution in places and transitions - Token information - Invariants (transition/place)
Analysis parameters	- Parameters for analysis such as expected number of tokens arrived, average firing time, average wait time, overall cycle time

The analysis of Petri net model can be divided into two parts. The first part deals with understanding structural properties of Petri nets such as liveness, reversibility and boundedness. These properties can be directly related to identifying issues such as bottlenecks and conflicts in sys-

tems. In order to perform such an analysis, the tool needs to have information such as adjacency matrix or reachability graph (Zhou 1999).This information can also be derived from structure and tokens at each place/transition.

The second part deals with simulation analysis to understand performance aspects such as mean time for firing of transition/place and average number of tokens at a place. In order to perform such analysis, the tool needs to know simulation environment parameters such as number of replications, replication length and transition/place firing distributions. The proposed extended PNML (xPNML) extends the PNML ontology to facilitate capturing the required information to calculate the desired performance metrics of the system being modeled.

3.2 Schema for simulation and analysis

Figure 5 provides an overview of xPNML generation. Unlike PNML, xPNML type definition is generated by Common Type Definition (CTD) and User Defined Module Definition (UDMD).

In PNML model, user designs a specific Petri net using Petri Net Type Definition (PNTD). Specific PNTD is referred implicitly by PNML meta model and has specific type definition for a particular model. For example, to design a stochastic Petri net, modeler should use PNTD for stochastic Petri net. Although there are many advantages associated with such an approach, it has a major disadvantage that it requires a specific PNTD before use. In addition, PNML's PNTD cannot contain user defined schema explicitly.

The reason for introducing xPNML is to overcome limitations associated with creations of PNTD. Although PNTD concept is small and compact, it needs to be fixed as a standard document before use. This makes broad use of PNML difficult. In xPNML, type definition is separated into common part and user defined part. Common part is based on xPNML meta model. We call it as a Common Type Definition (CTD). The other part is called a User Defined Module Definition (UDMD). As a result, the Type Definition document is not needed to be fixed as a standard document in xPNML.

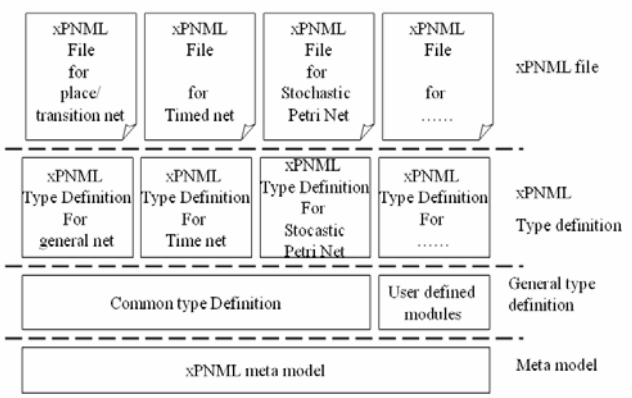

Figure 5: Overview of xPNML generation.

xPNML's type definition is made by CTD and UDMD. CTD is not changed schema/definition but UDMD is changeable schema/definition. These two definitions follow XML Data type Definition (DTD). So, xPNML type definition is generated by xPNML definition generator automatically. xPNML definition generator is a software module which can merge two DTD files into one DTD file.

This architecture provides a more powerful extension than that of PNML. In PNML, the software has to refer to each PNML Type Definition Document. So, before using it, each PNML Type Definition should be fixed as a standard. This late process for standardization results in the hesitation for selecting PNML format in many Petri net tools.

Figure 6 shows the process of generating xPNML file with CTD. To generate xPNML, software only has to refer to CTD standard document and optional UDMD. UDMD does not need to be fixed as a standard. So, xPNML is a more extensible structure than PNML. The xPNML type definition generator makes xPNML Type Definition with CTD and UDMD. In the process, xPNML type definition generator checks the validity of UDMD. User can generate specific Petri nets with their specific format, and xPNML generator generates xPNML file with the corresponding xPNML Type Definition.

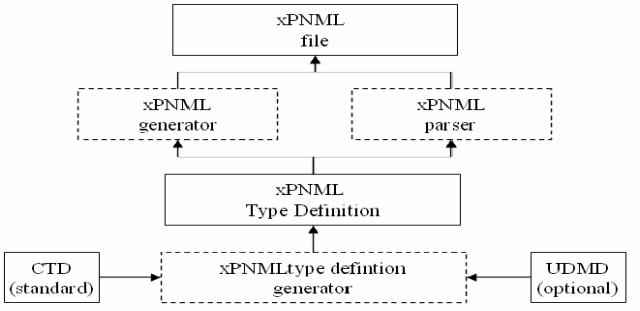

Figure 6: xPNML file generation mechanism in software

3.3 Modifications for simulation and analysis

xPNML extends PNML concept from design and modeling mechanism to simulation and analysis mechanism. Generally, a user's objective for using Petri nets are focused towards using Petri nets as a tool for modeling, simulation and analysis. Although PNML schema allows interoperability of Petri net models among different users, it does not allow specification of simulation information. Without such an information, shared information becomes more centered towards model sharing, but it can lead to misinterpretations as there is no common standard for comparing alternative Petri net models. Simulation information provides a benchmark against which multiple models can be compared. To prevent these misinterpretations and to account for simulation interoperability, xPNML includes schema for simulation and analysis.

Figure 7 shows the label entity in PNML model. The label entity consists of annotation and attributes (Billington et al. 2003) .

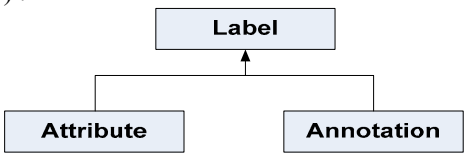

Figure 7: Label entity in PNML meta model (Billington et al. 2003)

Annotation describes information that is represented along with the Petri net node in graphics window. This includes information such as name, markings and stochastic information. The attribute information specifies the graphic property such as color and styles. The detail information about these entities is defined in Petri net type definition (PNTD). This increases the size of PNTD document and increases its complexity. Furthermore, most of the annotation information is stored in text format, which is difficult to use for simulation and analysis.

To overcome these limitations, the Label entity in xPNML is modified as shown in figure 8. The attribute node consists of general information, marking information and stochastic time information. The *General* information consists of names and inscriptions. The *Marking* information consists of number of tokens for places, or arc weights. The *TimeInfo* consists of stochastic information for firing of transitions. This information contains the type of distribution and its parameter values. This information will be used for simulation and analysis of Petri net models.

Figure 9 shows the simulation parameters added in Petri net meta file. The simulation parameters consist of *settings* and *analysis*. The *settings* consists of simulation environment parameters such as number of replications, replication length. The *analysis* component provides information about structural analysis and simulative analysis of Petri net model.

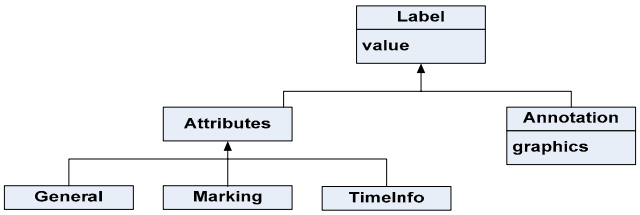

Figure 8: Label entity in xPNML meta model .

Figure 10 shows an example xPNML file with simulation parameters. A JAVA based software is being developed for editing xPNML file without professional xPNML tools. If the user knows xPNML schema, s/he can easily edit or modify the xPNML file. Also, the software supports validation of xPNML file. The left windows shows the structure of xPNML file shown in right window. For example, to change value of *number of steps*, the user can click value of *number of steps* entity in left window and this value is automatically highlighted in right window. This makes xPNML file editing simple and easy to update.

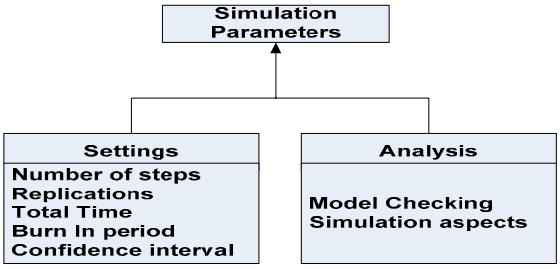

Figure 9: Simulation parameter entity in xPNML meta model.

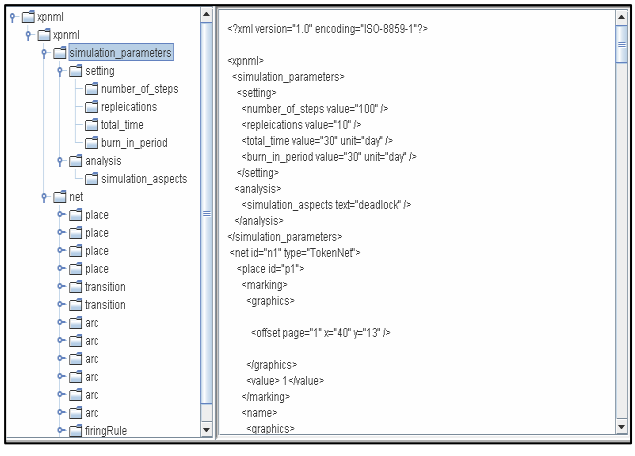

Figure 10: xPNML file with simulation parameters.

3.4 Modifications for modularity of Petri nets

For designing complex models and to simplify Petri net model structure, a modular concept is required. PNML uses reference objects : *Pages, Places* and *Transitions* for modular concept. An example of such a concept is CPN tools developed by university of Aarhus <http://wiki.daimi.au.dk/cpntools/cpntools.wiki>. The xPNML extended module concept is derived from PNML reference node concept and ARENA's modular concept <http://www.arenasimulation.com>. While PNML's node is just for merging other sub net, ARENA's modules have specific purposes. For example, packing modules in ARENA help users to design packing system easily. This helps in creating modules which can be easily imported and used in different Petri net models and thus help in saving development time.

The proposed xPNML meta model provides a schema for creating Petri net modules. These can simplify the structure of Petri net model and aid in easily understanding Petri net properties. In essence, it is similar to a template. For example, a machine template can represent all the states of a machine condition such as working, repaired or in failed state. There are certain advantages associated with the use of modular concept in Petri net models:

- The complex models can be simply represented using modular concept.
- The modular concept aids in reusability of small models across different Petri net models. For example, the designers can create templates of machines, and then use them in different Petri net models,

There can be two different types of modules: sub Petri net and sub object module. Sub Petri net module can be used to represent the control logic of small systems within a large system. On the other hand, the sub object module can represent certain objects or resources, such as machine within a Petri net model.

In xPNML, a module can be defined in user defined module definition (UDMD). UDMD is a type of XML Data Type Definition (DTD) and xPNML meta model has entities for linking UDMD to xPNML meta model. Figure 11 shows the schema for modular concept in xPNML. *ModularPetriNet* entity is for sub Petri Net modules and *ModularObject* entity is for sub objects.

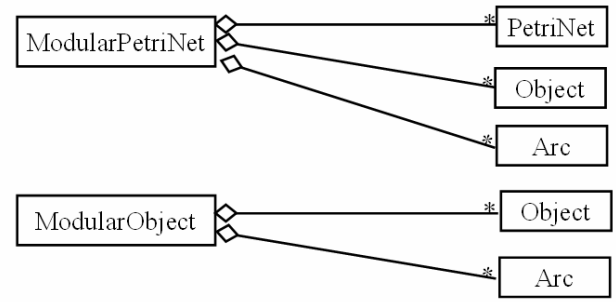

Figure 11: Schema for module concept in xPNML meta model

3.5 xPNML meta model

Figure 12 shows the xPNML meta model. The model includes the simulation, analysis and modular implementation for stochastic Petri nets. As mentioned in section 3.1, the label entity in xPNML meta model is modified with stochastic and marking information. Depending upon the type of stochastic Petri net (that is, Timed Place Petri net or Timed Transition Petri net), the time information can be added to each object (place or transition) and its parameters can be defined. The marking information associated with objects helps in the analysis of structural Petri net properties.

The simulation parameters are unique to each Petri net file and they define the simulation aspects for each model. This information can be used by tools for identifying the parameters that are to be evaluated (for example, average number of transitions fired, average number of tokens at each place) and simulation settings such as replications, replication length.

3.6 Tool design for xPNML

Table 2 enlists some of the key requirements for a good Petri net tool. The model should be able to import, create and export xPNML based Petri net models. The graphics editor for the tool should allow ease in model building and verification. The tool should allow support for various distributions in stochastic Petri nets. One of the limitations with current tools is that they do not allow user specific distributions. In addition, the tool should be able to verify structural properties of Petri net models and provide interactive simulation to understand the logic flow of information in the Petri net model. The tool should also allow discrete event simulation for performance evaluation of models under consideration.

Keeping these points in mind, a JAVA based xPNML design tool is presented here.

Table 2: Requirement for new Petri net tool

		Requirement and function
Common menu		- Interchangeable file format (xPNML format)
Graphics editor		- Ease of model creation - Modular method for Petri net design - Easy to input and output files
SIMULATION	Analysis	*<Supported distributions>* - Continuous distribution - Discrete distribution - User defined distribution *<Invariant analysis>* - Transition and Place invariant *<Structural analysis>* - Deadlock, reachability analysis
	Results	*<statistical representation>* - table representation - graph representation *<visualization>* - 2D based animation: token game animation - 3D based animation
Extra		- functions like workflow management

4 XPNML BASED PETRI NET DESIGN TOOL

Currently a new JAVA based Petri net design is being developed. The model will be fully compatible with xPNML structure and will meet all the requirements mentioned in section 3.

The new tool is being developed by modifying Petri Net Kernel (PNK) which is a JAVA based open source. <http://www2.informatik.hu-berlin.de/top/pnk/>. Figure 13 shows a snapshot of implementation stages for the tool. The tool allows creation of Petri net models and their export in xPNML format. The XML based interface allows visualization and modification of the structure of Petri net model. The final xPNML file can then be stored and used to visualize the changed Petri net in the JAVA based tool.

Currently, the tool supports xPNML based import, export and model creation. The tool also verifies the xPNML syntax and displays any error messages. In addition, the general xPNML based editor is developed based on Simple API for XML (SAX) technology. The tool is currently being modified to enable modular concept and simulation for stochastic Petri nets.

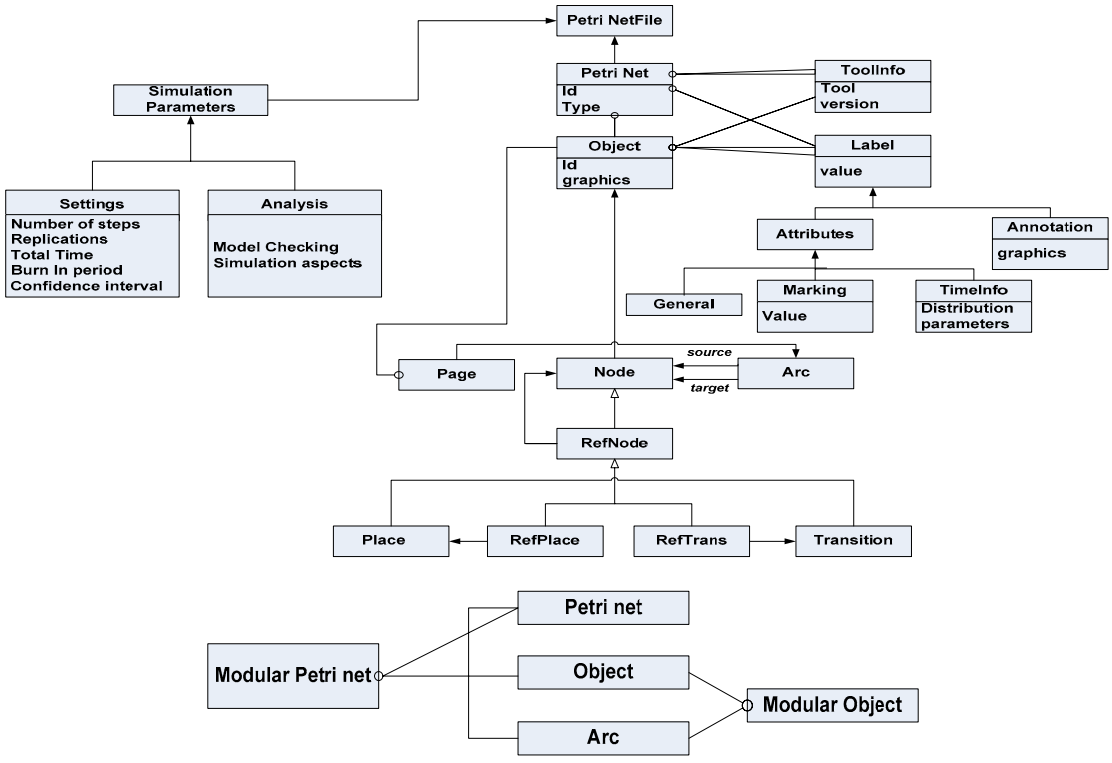

Figure 12: xPNML meta model

5 DISCUSSION AND FUTURE WORK

An xPNML based file format for sharing and modeling of Petri net model is developed. The format overcomes the limitations associated with PNML format and extends the structure to incorporate stochastic and simulation information. The xPNML meta model has a schema for specific modules. With these modules, we can easily generate complex Petri net models for specific problems. As compared to PNML, xPNML does not need to make and fix standard for type definition documents.

Currently a JAVA based xPNML tool is being developed, which will import and export xPNML based Petri net models. In addition, the model allows for checking of XML syntax and displays the Petri net structure. The model will be extended to simulate and analyze Petri net properties such as deadlocks and conflicts.

The current JAVA based xPNML tool can be extended for 3D visualization purposes. The transition firings in the Petri net can be mapped with 3D operations in a virtual environment. JAVA 3D can be a useful tool for such implementation.

The current approach has been tested for stochastic Petri nets and can be applied to colored Petri net models, which will increase the complexity of the model. Currently, there are only a few tools that support colored Petri nets.

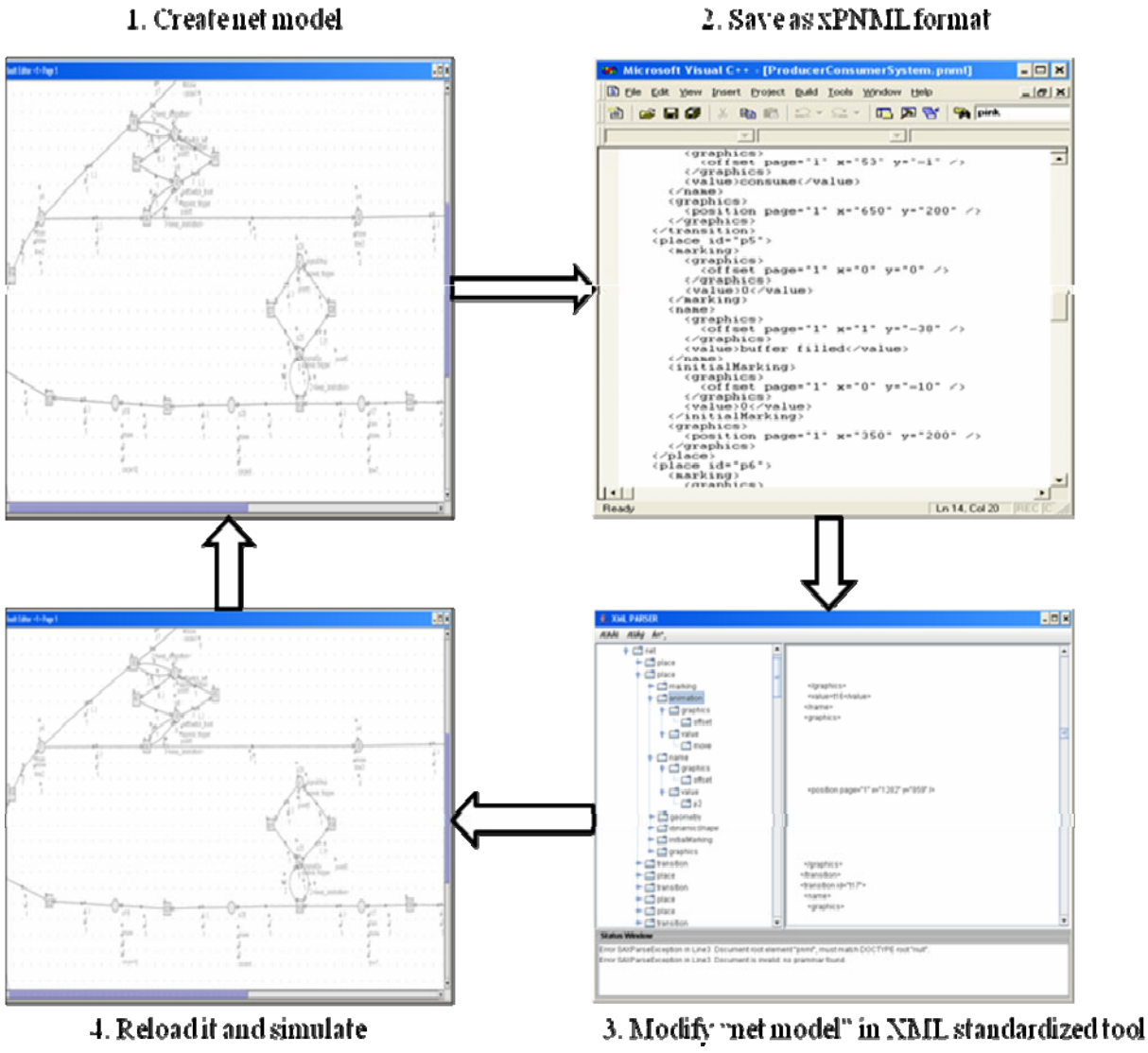

Figure 13: Snapshots of xPNML based Petri net simulator

REFERENCES

Al-Jaar, R. Y., and A. A. Desrochers. 1990. Petri nets in automation and manufacturing. *Advances in Automation and Robotics* 2: 153.

Baccelli, F., Balbo, G., et al. 1994. Annotated bibliography on stochastic Petri nets. *Performance Evaluation of Parallel and Distributed Systems - Solution Methods, O.J. Boxma and G.M. Koole, eds., CWI tract 105 & 106, Amsterdam, CWI* 105: 25-44.

Billington, J., Christensen, S., et al. 2003. *The Petri Net Markup Language: Concepts, Technology, and Tools*. Eindhoven, Netherlands. Available via <http://www2.informatik.hu-berlin.de/top/pnml/about.html> [accessed April 2, 2007].

Chung-Hsien, K., Han-Pang, H., et al. 2005. Separation model design of manufacturing systems using the distributed agent-oriented Petri net. *International Journal of Computer Integrated Manufacturing* 18(2-3): 146-57.

Jensen, K. 1997. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. 2nd ed., Springer-Verlag.

Kindler, E. 2006. Concepts, Status, and Future Directions, *Entwurf Komplexer Automatisierungssysteme*, 35-55, Germany. Available via. <http://wwwcs.unipaderborn.de/cs/kindler/Publikationen/copies/PNML-EKA06.pdf > [accessed June 22, 2007].

Murata, T. 1989. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE* 77(4): 541-80.

Zhou, M., and K. Venkatesh. 1999. *Modeling, Simulation, and Control of Flexible Manufacturing Systems—A Petri Net Approach*, World Scientific Publishing Company.

Weber, M., and E. Kindler. 2002. *The Petri Net Markup Language*. Petri Net Technology for Communication Based Systems. Available via <http://www2.informatik.hu-berlin.de/top/pnml/about.html> [accessed April 2, 2007].

AUTHOR BIOGRAPHIES

HYUNSOO LEE is an Doctoral student in department of Industrial and Systems engineering at Texas A&M University. He received his master's degree from POSTECH, Korea, and worked as a CAX/PLM consultant in SAMSUNG for 5 years. His research interests are in manufacturing simulation for lean manufacturing, artificial intelligent systems application and virtual manufacturing systems. His web page can be found via <http://ise.tamu.edu/varl>.

BIKRAM SHARDA is an Doctoral student in department of Industrial and Systems engineering at Texas A&M University. His research interests are in the application of Petri nets and Bayesian methods for uncertainty representation in manufacturing systems. His web page can be found via <http://ise.tamu.edu/varl>.

AMARNATH BANERJEE is an associate professor in the department of Industrial and Systems Engineering at Texas A&M University. He is an associate editor for the *International Journal of Services, Operations and Informatics.* His research interests are in virtual manufacturing, simulation, image processing, real-time video processing, augmented reality and human behavior modeling. His web page can be found via <http://ise.tamu.edu>.