Teaching Petri Nets Using P3

Dragan Gašević and Vladan Devedžić
FON – School of Business Administration
POB 52, Jove Ilića 154, 11000 Belgrade, Serbia and Montenegro
gasevic@yahoo.com
devedzic@galeb.etf.bg.ac.yu

Abstract

This paper presents Petri net software tool P3 that is developed for training purposes of the Architecture and organization of computers (AOC) course. The P3 has the following features: graphical modeling interface, interactive simulation by single and parallel (with previous conflict resolution) transition firing, two well-known Petri net analysis tools (Reachability tree, Matrix equations), as well as two new analysis tools (Firing graph, and Firing graph) developed for learning purposes. The special aspect of the P3 is the XML/XSLT-based support for model sharing with the following Petri net tools: DaNAMiCS, Renew, and Petri Net Kernel (PNK). This paper also gives overview of the AOC course, and compares students' outcomes in the AOC course when they used the P3, with the previous course outcome when the students did not use P3. Finally, the paper shows how teachers (i.e. we) and students perceived P3's features.

Keywords

Petri net teaching, hardware teaching, simulation-supported learning, XML-based interoperability

Introduction

Petri nets are mathematical formalism intended to be used for modeling, simulation and analysis of different kinds of systems (Peterson, 1981). In computer science Petri nets are used for modeling a great number of either hardware and software systems, or various applications in computer networks. A special advantage of Petri nets is their graphical notation, which reduces Petri nets learning time and simplifies their use. Hence Petri nets are used for teaching numerous concepts in computer science (Jeffrey, 1991). We should note that some authors (Denning, 2000) say that experimental methods in computer science have frequently lead to important developments in both theory and practice. Constructing models and performing simulation on them, are the core issues in experimental methods (Desel, 2000).

Considering these Petri net features within a course of Architecture and Organization of Computers (AOC) at Military Academy in Belgrade, one could notice that they are used for modeling different concepts included in this course: control unit organization, arithmetic-logic unit organization, organization of buses, elements of multiprocessor communication as well as hardware modeling and measuring performance of computer systems (Stallings, 2000). We use the Upgraded Petri nets (Štrbac, 2002) that are developed in order to enable use of Petri nets for hardware modeling, as well as to provide modeling at register level. During the realization of the AOC course, a need to use adequate software has occurred.

Firstly, in this paper we give an overview of the AOC course curriculum and emphasize how we use Petri nets in this course. Then we present the P3 – a Petri net tool developed for teaching Petri nets within the AOC course (Gašević & Devedžić, 2003). Our main intention is to provide better software support for Petri net's education since we have a limited number of classes for introducing Petri net concepts. The basic idea of this approach is to provide simulation-supported learning Petri net concepts, as well as speeding-up Petri nets learning process to the level required for their use in the AOC course. The P3 has XML-based model sharing capabilities. Actually, the P3 uses the Petri Net Markup Language (PNML) – on-going Petri net community effort for a standard Petri net model interchange format (Billington et al., 2003). We have developed three eXtensible Sylesheet Language Transformations (XLST) in order to transform PNML that P3 exports to the formats of the following Petri net software tools: DaNAMiCS, Renew, and Petri Net Kernel. We have also implemented a XSLT that converts PNML to the Scalable Vector Graphics (SVG). In that way we enable teachers to create Web-based Petri net learning materials using the P3 (i.e. the P3 is a kind of authoring tool). We analyze the P3 tool through subjective students' and teachers' experiences. Students' opinions were obtained from a questionnaire students had at the end of the AOC course.

Course of Architecture and Organization of Computers

P3 tool was developed with an aim to be used within the AOC course. In the previous course realization forms we have found that, the tool support they were based on, was inadequate since the curriculum of the AOC course has limited number of classes planed for introducing Petri nets. Besides, we wanted to parallelly introduce computer architecture concepts to our students and to show them Petri net models of those concepts. The AOC course is for the 4th year undergraduate students of computer engineering. This course is a natural extension of the previous course in computer architecture – Basics of computer engineering. Besides computer engineering, our students have also good knowledge of electronics, digital electronics, and software engineering. While creating curriculum for the AOC course we were following recommendation given in (Cassel et al., 2001).

In Table 1 we show the AOC course structure. Firstly, we introduce Petri net concepts since we want to use Petri net in all course lectures. In the Petri net introductory lecture we present Petri net definition, Petri net graph, Upgraded Petri net definition, and common Petri net analysis tools (e.g. reachability, invariants). Then, we lecture computer architecture concepts and give Petri net models to explain formal verification for these concepts. Our main focus in Petri net modeling are exercises where students solve computer architecture problems using Petri nets. They need to use different Petri net analysis tools to verify their models. In that way, they learn practical value of having formal verification tools in hardware design. But, very often it is not enough to use only one Petri net software tool to obtain satisfactory solution. Further more, as we have already mentioned, we should have Petri net tools that can be used for learning and understanding Petri net analysis tools. Of course, Petri net tools should have advanced simulation and analysis features, since in the AOC course we also show a few real-world examples (Wilson et al., 1993). In that way we avoid over-simplifying instruction. In the next section we explain Petri net tools we have used in the previous AOC course realizations as well as mention obstacles that guided our decision to develop a new Petri net tool.

Table 1. Structure of the course Architecture and organization of computers in which P3 is used

Lecture	Lecture tag	Number of hours		Petri net models
		Lecturing	Exercises	
Petri net introduction	–	2	–	–
Processor architecture	1	5	5	2
Processor implementation	2	5	5	2
Arithmetic	3	6	5	2
Memory organization	4	6	5	2
I/O subsystem organization	5	6	5	3
Busses	6	3	2	3
Multiprocessor systems	7	3	2	4
Vector and matrix processors	8	3	2	4
Elements of distributed systems	9	3	2	5
Modeling and measuring of computer performances	10	3	2	3

Current Petri Net Tools

There are many Petri nets software tools (PNTDB, 2003). Three of them – DaNAMiCS, Renew and Petri Net Kernel (PNK) – we have used in the previous course realizations. Here we describe their features as we have perceived them during realization of the AOC course. The graphical user interface for modeling is a common feature of these three products. Although these tools offer extensive possibilities for Petri nets modeling, they do not provide detail explanations if it happens that a model contains a syntax error (e.g. arc between two places). With such notifications, learning process would be more effective. Concerning the simulation support, the three programs have interactive execution based on a single transition execution. However, a parallel execution would be useful as a teaching support, since it could help students to understand principles for defining priorities in conflicts states, as well as it could provide explanation why some simulation step can not be made although the marking state is fulfilled. Only DaNAMiCS implements analysis tools, but it does not have the capability to interactively map results to the observed model. That restrains the possibility of using DaNAMiCS for learning Petri nets analysis algorithms.

From these observations we concluded that for our purposes, it would be necessary to develop a new application. The following facts support this conclusion:

1. None of the three Petri nets software products implements the Upgraded Petri nets that are suitable for hardware modeling at the register level.
2. It is necessary to implement a new tool for model analysis that will be suitable for understanding model execution and conflict state analysis in order to eliminate them. A Firing graph has been developed for these purposes. Its details are presented later in this paper.
3. Although there are different publicly-available Petri nets software tools, we have implemented a new one in order to support teaching requirements of the AOC course from the very beginning.
4. We have not found any Petri net software tool that is extended with a support for transformation of its model format to the formats of other Petri net tools. For that purposes we suggest the XML/XSLT-based approach.

The P3 Architecture

The P3 is developed according to the facts we have given in the previous section. Further more, since the P3 is based on the PNML concepts, it is compatible with the PNML. We identified main parts that are related to the Petri net use. These parts are as follows:

- *Petri net structure* – The central part of the Petri net structure is a Petri net that consists of the Petri net basic concepts: places, transitions, and arcs (Murata, 1989). Petri net places and Petri net transitions can be generalized as Petri net nodes. Important part of the Petri nets that pertains their structure is marking (token current state of a Petri net), and initial marking. Although these Petri net concepts are not formally a part of the Petri net structure, they could be implemented as a part of the Petri net structure (places). Since we implement software tool that supports Upgraded Petri nets, we extend the Petri nets structure with additional concepts. That means:
  - a Transition Function (TF) is attached to each transition. TF is performed when a transition is fired, e.g. ADD, OR, SUB, EQ (equal), etc. The Transition Firing Level (TFL) is connected with the assigned TF. Timing function is also attached to a transition – the probability that a certain transition will be fired in a certain period of time.
  - each place has two attributes: X – a real number attribute that defines a place state (that can be treated as a register state), and Y – an integer number attribute that is related to the order in which a place will be analyzed when a transition is fired.
  - arc type – normal, inhibitor.
- *The Petri net graph* - is closely related to the Petri net structure. It can be said that the Petri net graph is a graphical notation for the Petri net structure. Thus, the elements of the Petri net structure and the Petri net graph can be implemented together, or dependent on each other. We use term modeling for creating a structure and a graph of some specific model.
- *Petri net simulation* - means firing (execution) of enabled transitions. A simulation changes current marking of a Petri net, and that causes appropriate changes of the Petri net graph. We wanted to implement two different modes of simulation: through parallel execution of all transitions that can fire with a previous conflict resolution; through single execution of a transition that can fire. Each mode of simulation changes a Petri net structure and interactively maps those changes to the Petri net graph.
- *Petri net analysis tools* - there are many well-known Petri net analysis tools, e.g.: Reachability Tree (Peterson, 1981), Matrix Equations (Reisig, 1985) etc. We have also introduced new analysis tools that are appropriate for teaching purposes: Fireability Tree (Štrbac, 2002) – simplified reachability tree, Firing graph (Gašević et al., 2003) – executes the specified number of the simulation steps, draws firing graph of the simulation states (marking), and annotates the conflict situations. We wanted to enable interactive mapping from the analysis results to the model.
- *Petri net model sharing* – that feature should enable model exchange with different Petri net tools, as well as sharing models on the Web. In that way more software solutions could be used for observing the same model. Currently, the XML-based solutions are most common. Especially important is the PNML, a general proposal of Petri nets universal markup (Weber & Kindler, 2003).

According to the identified parts we have created and organized Petri net classes that constitutes the P3's architecture, shown in Figure 1. The Petri net class organization is shown on the left in Figure 1, whereas the supported formats are on the right side. Elements of the Petri net classes are organized according to the previous analysis.

P3's architecture parts that relate to the model formats are also based on the PNML. The PNML extension (UPNML) has been made for Upgraded Petri net markup. We have extended P3 model sharing capabilities with

XSLTs that transform the P3's output format (i.e. PNML) into formats of Petri net tools we have already mentioned: DaNAMiCS, Renew, and PNK. Details of these sharing features as well as experience with their application in practice are given later in this paper. We have also implemented an XSLT for transforming PNML (as well as UPNML) documents to the SVG – the W3C's standard for 2D vector graphics (Ferraiolo, 2001). In this we made possible creation of Web-based Petri net teaching materials using the P3. Additionally, if we empower this SVG format with annotations (e.g. RDF) and Petri net ontology (Gašević, 2004), we will be able to restore Petri net model semantics for the SVG representation.

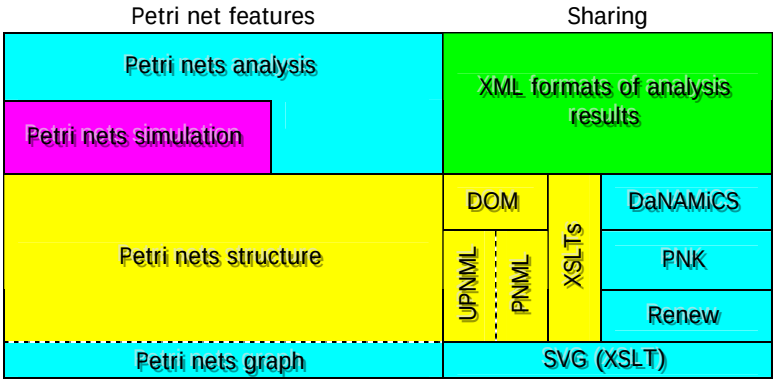


Figure 1. P3 architecture: class organization and supported XML formats

P3 - User Features, and Educational Petri Nt Software

P3 has been developed as a standard Document/View application. It is implemented in C++ using the Microsoft Foundation Classes (MFC) library. The Xerces XML parser version 2.3.0 (http://xml.apache.org/xerces-c/index.html) has been used. XML documents are parsed using the Document Object Model (DOM).

Figure 2 shows an example of P3 screen. The workspace is divided into two parts: on the left side there is a navigation tree of Petri net objects; on the right side, there is a document review of the Petri net graph which is being operated at the moment. Names of all transitions and places of the active Petri net are listed in the left part of the window. Through selection of a particular item in the navigation tree, an appropriate Petri net object will be selected. Features of the selected object, such as name, transition function, place marking, can be changed. This tree has proven to be very useful in practice, since the students got accustomed to the similar user interface organization during their previous software engineering courses (e.g. Rational Rose). It is also possible to write annotations for some objects of the graph shown on the right side of the window. The object selected in the navigation tree can be deleted. In the right part of the window, the graph of the active Petri net is drawn. The modeling function performs in this window. The results of the simulation as well as a single simulation step are also shown.

While places are represented as circles, transitions are in the form of rectangles that become two times wider after a transition is fired – this is useful for learning and better detection of fired transitions. Net nodes are linked by arcs, which were outlined using the Bezier curves. Inhibitor arcs terminate with a circle, whereas arrows at the arc ends mark the direction of ordinary arcs. There is no limit in the number of breaking points. To link two nodes it is necessary to select (using the mouse) a starting node (place or transition), then the arc breaking points and finally, a target node of the Petri net. If a user tries to link nodes of the same type (for example place and place), such linking would be rejected and a dialog would appear giving the information about syntax error that has been made. This P3's feature is useful in the process of learning the Petri net.

Simulation of developed models can be done in two ways:
1. by parallel execution of all transitions that can fire with a previous conflict resolution;
2. by single execution of a transition that can fire.

The first type of simulation is performed in response to a selection of an appropriate menu option (Analyze → Execute Parallel Step), whereas the second one is performed after selecting (using mouse click) a transition that is intended to execute. Both simulation types are performed interactively i.e. their execution changes a marking state of the analyzed net, and the change is automatically reflected on the Petri net graph view in the P3.

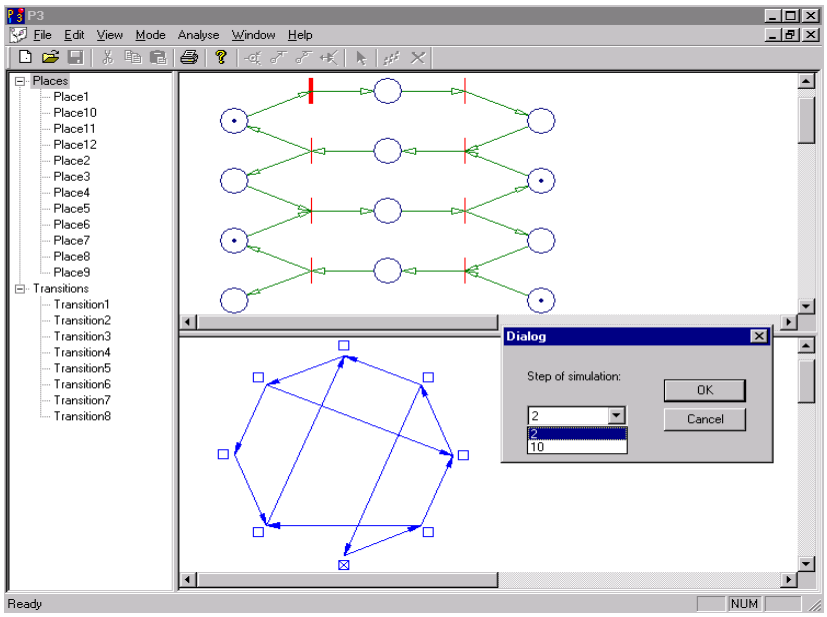


Figure 2. P3's graphical user interface and Firing graph on the example of pipeline model

P3 implements the following model analysis tools: *Firing Graph, Reachability Tree, Fireability Tree*, and *Matrix Equations*. To choose one of those tools, one should use program menu option Analyze->Analyze with, and then select an option with the name of an appropriate analysis tool.

After a calculation with a chosen analysis tool has finished, the right side of the P3 window splits into two horizontal parts. Result of the computation is drawn in the lower part. This layout was made to enable parallel observation of the model and the analysis tool during the learning process. When analysis tools – Reachability tree, Fireability tree, and Firing graph are used, this layout makes possible interaction between a model and the result of an appropriate model analysis method. This paper describes the way P3 presents the Firing graph and performs interaction with the analyzed model. Details of other tools have not been described since the length of this paper is limited.

The Firing Graph has been developed to present the sequence of parallel fireable transitions during the execution of a specified number of simulation steps. In this way one can observe the simulation procedure and conflict situations. Consequently, this tool is convenient for realizing the need of Petri net model analysis, and that is why students should use it in the first phase of learning Petri nets. A set of parallel fireable transitions is marked by □ in the P3 program package and it includes information about simulation step that the system reached in certain state. That information is obtained using a mouse click to select a desired state. A dialog shown in Figure 2 appears, with iteration numbers in which a net in that state is. The initial set of the parallel transitions that can fire is presented by ⊠. If there is a conflict in some state of the parallel transition execution, then that state has a mark □⊢×. The mark X can be on any side of the rectangle that represents a set of the parallel fired transitions, considering that it is out of the circle which can go round all states the system is during the simulation. The transition to the next state is marked with a directed line. In case transition is performed in one and the same state in the firing graph, such set of the parallel fired transitions is marked with ⊡. The presented implementation of this tool enables easier learning of the main algorithm assumptions. An important factor that makes learning easier is interaction between analysis results and the observed model.

P3's XML/XSLT-based Model Sharing

We based P3's format for model sharing on the PNML (Gašević et al., 2003). This P3's format extends the PNML XML Schema definition that is downloaded from (PNML, 2003). We have made the PNML extension in the form of the XML Schema, and for that purposes we used the UML profile for modeling XML Schema (Carlson, 2001). In that way, we obtained the PNML documentation as well as documentation of extensions we have made. Furthermore, this solution is independent of the target XML schema definition language (XML Schema, TREX, RELAX NG, etc.).

We have already mentioned that we have extended the P3 with XSLTs that convert the PNML format into the DaNAMiCS, Renew, and PNK formats. These XSLTs are omitted here due to their length. The transformation principle that was used is illustrated in Figure 3: a P3-generated PNML-based model is the input to the XSLT processor, and it is converted using the XSLT corresponding to the target tool. DaNAMiCS was selected because we wanted to share a PNML-based model developed in P3 with a tool that uses an ordinary text format. Renew was suitable for showing the exchange of P3 models (PNML-based) with a tool that uses another XML-based format, and PNK enabled interchange between two different PNML-based tools. In this paper we give only short overview of our experiences with this model sharing principle, while details can be seen in (Gašević et al., 2003). From the teaching point of view, one can discern the advantages obtained using multiple tools in model observation: DaNAMiCS (analysis tools), Renew (rich graphical environment), and PNK (an infrastructure for Petri net tool development that is convenient for extensions). Of course, there are few constraints of this sharing since none of the analyzed Petri net tools implements Upgraded Petri nets. Recently, we have started to use the PIPE Petri net tool (https://sourceforge.net/projects/petri-net). Our first experience shows that we do not need to perform any additional transformation of P3's PNML models in order to import them in the PIPE tool.

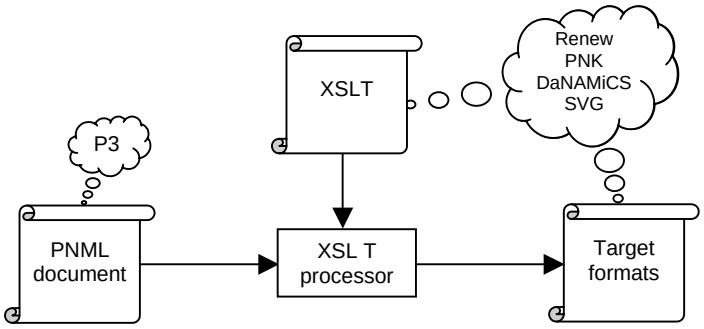


Figure 3. The principle of XSLT-based model conversion from P3 to another format

DaNAMiCS

DaNAMiCS tool is freeware and can be downloaded from http://www.cs.uct.ac.za/Research/DNA/DaNAMiCS/. DaNAMiCS' important advantage is its support for a number of analysis tools, such as matrix invariants and transition matrices, structural analysis, as well as some simple and advanced performance analyses. Due to the fact that DaNAMiCS has more analysis tools than P3, it can improve model analysis for Petri nets developed with P3 when a P3-based model is converted into DaNAMiCS format. Format that DaNAMiCS uses for model import (*File* menu, *Import net* option) is *wam*. Using XSLT, we have converted some P3 models into *wam* format so that they can be imported into DaNAMiCS. However, it is not possible to use XML/XSLT-based approach to convert models developed in DaNAMiCS into some other format, since such models are described as text-based files.

Renew

The Renew tool for Petri nets development can be downloaded from http://www.renew.de. It supports several Petri net dialects: object-oriented Petri nets, high-level Petri nets, P/T nets, and timed Petri nets. Its advantages include: support for synchronization channels, an advanced communication mechanism; support for modeling object-oriented concepts; a number of supported arc types; rich graphical environment (Kummer & Wienberg, 2000). Unlike DaNAMiCS, Renew uses XML to overcome the problem of model exchange with other Petri net tools. XML documents containing Petri net models can be described using a DTD (Kummer et al., 2001). Such DTD is defined starting from the same assumptions that underlie PNML, and as a consequence those DTDs have common elements (net, place, transition and arc). We have developed an XSLT that converts a PNML document into Renew XML format.

Petri Net Kernel (PNK)

PNK is publicly available from http://www.informatik.hu-berlin.de/top/pnk/download.html. PNK is not just another Petri net development tool – it provides infrastructure for building such tools. It is not focused on a specific Petri net dialect; it is possible to use PNK with Petri net dialects with specific extensions (Kindler & Weber, 2001). Publicly available version implements the following Petri net dialects: P/T nets, High-level Petri

nets, Timed Petri nets, Bag-based Petri nets, echo nets, and BlackToken nets. PNK also implements graphs, and ghs graphs. The built-in set of dialects and graphs is extensible with new kinds of nets. PNK uses PNML for annotating documents containing Petri net models. It considerably facilitates model sharing between P3 and Petri Net Kernel. However, there are differences in definitions of PNML used in these two tools, hence model sharing between them required building an XSLT as well.

P3 Supported Lessons

In this section we show P3's usability for teaching computer architecture, as well as for model sharing. Firstly, we give an example of a microprogram machine instruction. In order to show how we share P3-developed models with other Petri net tools, we use the well-known synchronization problem – Dining Philosophers (Silberschatz et al., 2003).

The First Microprogram Modeling Lesson: the BIFE Instruction

The control unit implementation techniques (e.g. microprogramming) are a part of AOC course. Here we show the first lesson in using both Petri nets and P3 for microprogram modeling. The example refers to the execution phase of the BIFE (Branch if equal) instruction, which is executed in a simple process model described in (Chu, 1972). Since this model refers to the real system modeling, it is introduced after basic concepts of Petri nets and P3 software, have been covered. The machine instruction compares values of registers A and B, if they are equal, it branches off and increases, in execution phase, the value of the program counter by 8, otherwise it does nothing.

The execution order of micro-operations of this instruction and their representation by Petri nets are described later in the paper. Description order is the same as the one used for their teaching. The model of this microprogram is shown in Figure 4. The X attribute of the Upgraded Petri net represents a state (content) of a component that is used in the microprogram (registers, buses etc.), whereas the Y attribute defines the order of the transition input places (i.e. their X attributes) during function calculation (addition, multiplication). Presentation of the microprogram instruction steps and description of the Petri net elements related to the given microprogram, are given later in the paper.

1. *Transfer of A register's content to the left bus.*

   The register *A* is modeled in the Petri net as a place with the same name (A), whereas the *Lbus* is the left bus. In the Petri net this step represents the execution of the *COPY* transition.

2. *Transfer of the first complement of the B register's accumulator content onto the right bus, parallel adder activation so that the adding is performed with C=1 (we calculate A-B, where B is presented in second complement notation).*

   The register *B* is represented as a place with the same name (B), whereas the place *RBus* models the right bus. The complement of the B register's content is modeled by multiplying (the *MUL* transition) the B's content by -1 (place *Const2*). The adder activation is represented by the execution of the *COPY1* and *COPY2* transitions and token generation into the *P3* and *P4* places, which represent the adder entries. The adder performance is represented by the *ADD* transition, whereas the addition result is in the *SUM* place.

3. *Output of device for zero check (ZT sets) sets flag Z.*

   Checking if the result of the addition (the *SUM* place) equals zero (the *EQ* transition) or not (*NEQ*).

4. *Storing the address into the microprogram address register*

   In case that the register's content is not equal zero, the *NEQ* transition is fired and the token is generated in the *New instruction* place, which means that it is possible to obtain a new instruction for the execution.

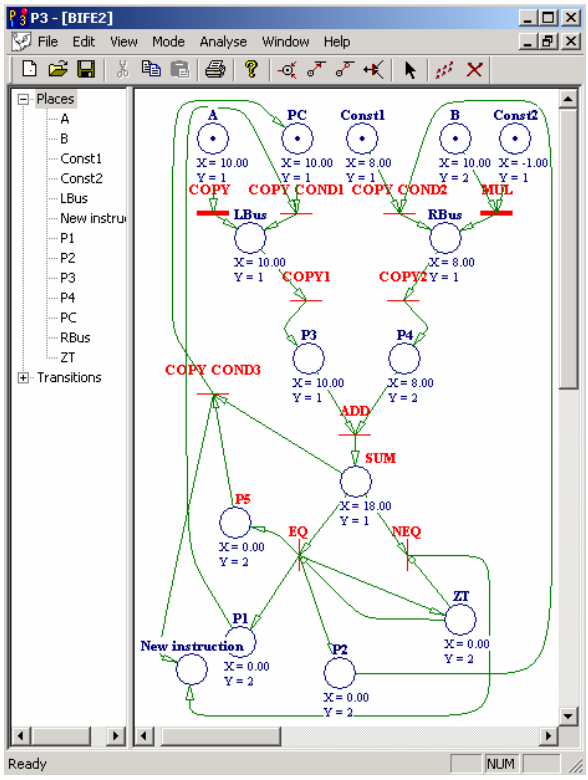


Figure 4. A microprogramming lesson: the BIFE instruction

5. *If condition A=B is fulfilled the content of the PC is put on the left bus*

   This step fits the execution of the EQ transition (SUM was 0), and after execution is done, tokens are generated into P1, P2, and P5 places. The P2 place serves to signal firing of the COPY COND1 transition because the place that models the PC already has a token. The PC content is copied on the left bus (LBus) through execution of the COPY COND1. A token in the P5 place should enable copying of computed result into the PC (a new value).

6. *Setting the constant (c=8) on the right bus.*

   The constant 8 is represented as the Const1 place. This step is modeled as execution of the COPY COND2 transition and copying of the Const1 content into the RBus (the right bus). The token has previously been generated into the P2 through execution of the EQ transition and the COPY COND2 becomes fireable.

7. *Parallel adder activation (adding PC + 8).*

   The adder activation is represented as execution of COPY1 and COPY2 transitions and bringing left and right bus content (LBus and RBus places) to the adder entries (P3 and P4). Adding is represented as execution of the ADD transition followed by placing the result in the SUM place in which this token is also generated.

8. *Result transfer from the main bus into the PC.*

   This operation is modeled as execution of the COPY COND3 transition and copying SUM content into the PC. The token is also generated into the New instruction place, which is a signal for obtaining a new instruction.

Pedagogically, this example is suitable for illustrating the use of the Firing graph, as well as for understanding approach to its simulation-based calculation. Namely, in the first iteration of the Petri net shown in Figure 4,

there is not any output arc from the EQ transition to the ZT place. According to this, when the content of A and B registers are equal, the PC content is increased by 8 (transition ADD). Consequently, there is a conflict in the SUM place (a token for the NEQ and COPY COND3 transitions, where the NEQ should not be fired again). This simulation flow is illustrated by the Firing graph from Figure 5a, which has one conflict state. If there is the arc between the EQ and ZT, then the conflict is eliminated (Figure 5b).

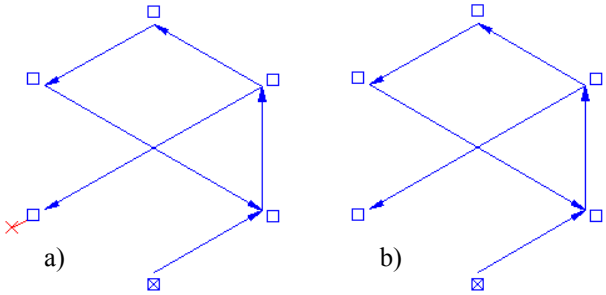


Figure 5. Studying the Firing graph: a) with conflict, b) without conflict

P3 Interoperability

Distributed and multiprocessing computer architectures are an advanced part of the AOC course. In this course part students can see full Petri net potentials for system modeling since Petri nets have different analysis tools, for example, that can detect deadlocks. In the AOC course, we firstly, introduce distributed hardware concepts and synchronization mechanisms using the examples of well-known problems (e.g. Dining philosophers, Producer/Consumer, etc) (Peterson, 1981; Murata, 1989; Silberschatz et al., 2003). Then, we show how these concepts can be analyzed using both Petri nets and P3. At this point, we mention students that P3 has a limited number of analysis tools. For instance, the P3 does not support the following analysis: boundedness, liveness, T- and P- invariants, different time analysis, coverability graph (Murata, 1989), etc. However, these can be done using DaNAMiCS tool. Hence, we emphasize P3's model interoperability. In Figure 6 we show the Dining philosophers Petri net modeled in P3. Also, this Figure shows reachability tree of this model that can be used for detecting deadlocks. Students can use P3's reachability tree not only to analyze a model, but also to understand this analysis tool since P3 provides result mapping from a tree (i.e. its nodes) to a Petri net.

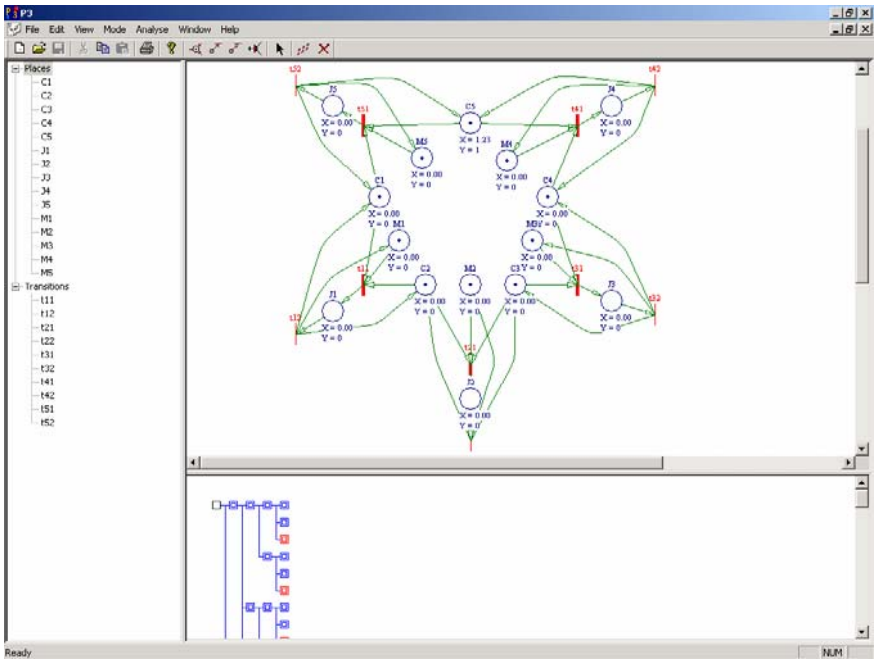
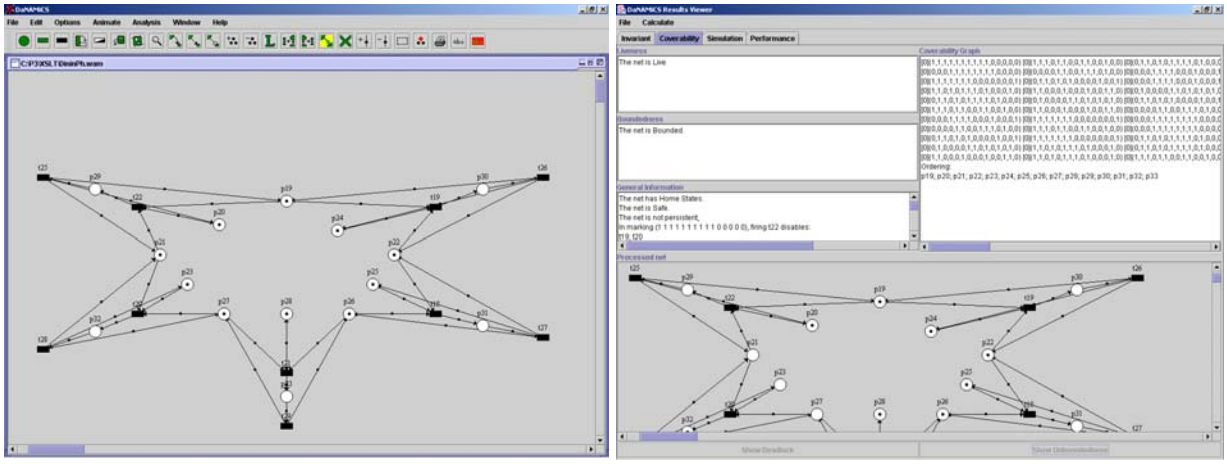


Figure 6. Modeling distributed systems using P3: problem of Dining philosophers

In Figure 7a we show the same Petri net model of Dining philosophers that we have exported from P3, and imported in DaNAMiCS. Figure 7b depicts a part of DaNAMiCS' analysis capabilities: liveness, boundedness, coverability graph, as well as annotation of deadlock and unboundedness. Of course, we should note that Petri

net models, which P3 shares with DaNAMiCS, can not be created in Upgraded Petri nets since DaNAMiCS does not support this Petri net dialect. There are similar limitations for model sharing regarding the Upgraded Petri nets between the P3 and Renew, PNK, and PIPE.



a) b)

Figure7. P3 and DaNAMiCS model interchange: a) exchanged model of the Dining philosophers from Figure 6; b) a part of DaNAMiCS' analysis results of the exchanged model

Analysis and Evaluation

An important, both practical and research topic in the area of Intelligent Educational Systems (IES) is that of evaluation strategies and performance measures (numeric) that could help quantify and rank a given learning environment and assess how different learning environments compare to each other (Devedžić, 2003). Note that evaluation is concerned with IES performance and its decision-making capabilities, as opposed to assessment, which is about the effectiveness of the system based on measuring learners' outcomes (Gilbert et al., 2001). Evaluating IES is difficult because the underlying theories are either new or still under development, and there is no widespread agreement on how the fundamental tasks (student modeling, adaptive pedagogical decision making, generation of instructional dialogues etc.) should be performed (Mitrović et al., 2002).

Although P3 tool is not classical educational systems (i.e. IES) it can be considered as a part of learning equipment. In this paper we analyze P3 in order to show: P3's impact on students' outcomes at the end of the AOC course, students' subjective evaluation of the P3 tool, and teachers' (i.e. ours) subjective observation of P3's benefits to the course realization. It is important to note that each year we have only one group of students in summer semester. In this paper we show results we got in realization of the AOC course last year (2003). The group consisted of 55 students. We have compared this group's results with results of the group that attended the AOC course a year before (2002). That group consisted of 50 students.

Students' Outcomes

We tested our students after each lecture in the AOC course. These tests were not performed on computer, but in the written form. In the figure 8 we give comparative overview of students' results when they did not use P3 during lectures, with the results obtained when students did use P3. One can see that the average student score is higher with P3 (77.27% average score of 55 students – these results are for the course in 2003) than without it (75.19% average score of 50 students – results from 2002). It is interesting to note that main improvements were in the following lectures: 7 (Multiprocessor systems), 8 (Vector and matrix processors), and 9 (Elements of distributed systems). Such results were expected since the main purpose of Petri nets is to model and simulate distributed and parallel systems and processes. In some other lectures (e.g. lectures 3 and 4) one can note lower score with P3. We should point out that in these two tests our main focus was not measuring Petri net skills. In this moment, we are not yet able to draw any general conclusion about P3's influence on the AOC course. In order to make a more general claim we should observe students' results in a longer time period (e.g. five years).

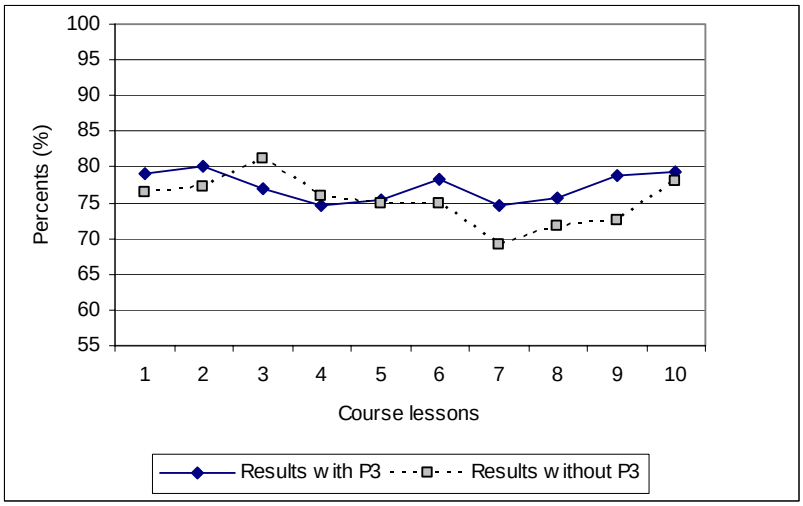
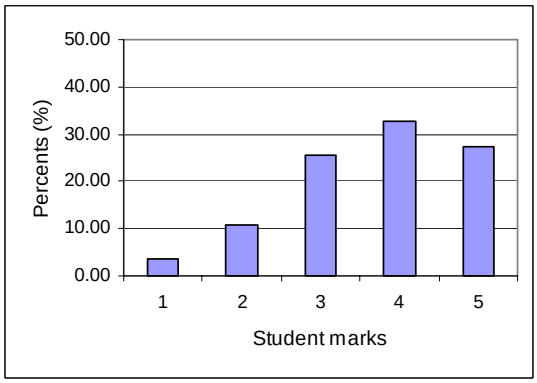


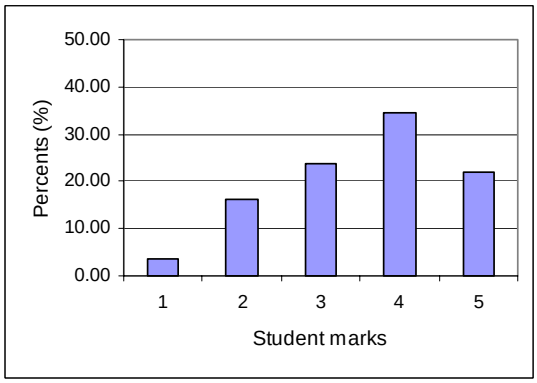
Figure 8. Comparative overview of student assessment in the AOC course: student results without P3 (First year) and results with P3 (Second year)

Students' Subjective Evaluation

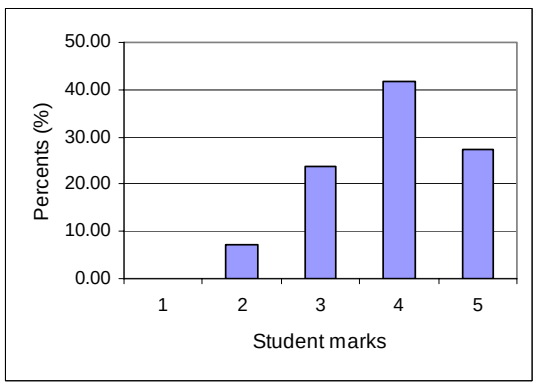
Here we present a summary of the students' answers to the user questionnaire. The students filled out a questionnaire at the end of the AOC course in which they used P3. The purpose of this questionnaire was to evaluate students' perception of the P3. The questionnaire consisted of 10 questions, each one based on the Likert scale with five responses ranging from very much (5) to not at all (1). Additionally, at the end of the questionnaire we provided a place for free form students' comments. We based our approach to evaluation of students' perception of the P3, on experiences shown in (Mitrović et al., 2002).



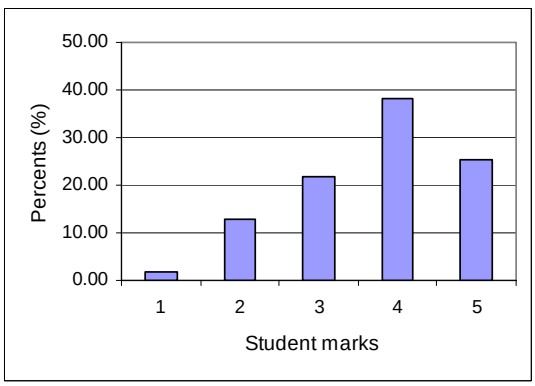
a) Do you find the interface easy to use?



b) Do you think P3's analysis tools useful?



c) How do you mark P3's interchange facilities?



d) Did you find P3 tool useful for understanding Petri net concepts?

Figure 9. Students' responses from the questionnaire at the end of the course (1 – Not at all, 5 – Very much)

In Figure 9 we give students' answers to some questions. We believe that shown answers can depict students' perceptions of P3's features in regard to our starting goals. It should be noted that P3's model sharing facilities (Figure 9c) are best marked. Actually, majority of students' comments regarding this feature suggested that it would be useful to empower a future P3 version with bi-directional interchange ability (e.g. P3 should have capabilities to import models from DaNAMiCS and Renew). Students remarked that the main P3's drawback is absence of structuring mechanisms since all real-world systems use Petri net concepts. Furthermore, they gave lower marks for P3's user interface (Figure 9a). All students had consistent agreement that P3's analysis tools (Figure 9b) and P3 tool itself (Figure 9d) are useful for understanding Petri net concepts.

Teachers' Subjective Analysis

We have noticed the following advantages P3 supported learning process:

- Students learn Petri nets' syntax while using this software, since P3 does not allow breaking of the syntax rules. This refers to the situation of connecting nodes to networks. For example, P3 does not allow a connection between two transitions. P3' dialogs that notify user about inaccuracies that caused an error, have proved to be especially useful.
- Execution rules of Petri nets can be learned through interactive execution of simulation. Students can understand simulation of simple models without problems. However, with complex models, ability to use a software that interactively changes the state of the Petri nets graph (marking) after the execution of the simulation steps, proved useful.
- Implemented analysis tools make basic ideas of their algorithms easier to understand. This is possible to accomplish by copying the results onto the analyzed model. We have already illustrated this process with an example of the Firing graph. The Reachability Tree and the Firing Tree are used in a similar way.
- A need for system modeling, precise definitions of design requirements and their adequate software support are obvious.
- Students understand advantages of using both simulation and software simulation tools, with the aim to cut the costs of hardware systems' development and increase system development productivity through exploitation of software modeling support. In this way students understand the need for software support for complex systems modeling, since without such support it would be difficult to perform required model analysis.
- P3 software makes verification of ideas possible even without physical implementation. The use of the P3 for these purposes, as well as its use for the detection of conflicts in the model are shown in the Firing graph in Figure 5.
- Students understand importance of model sharing for Petri net supported development. With model sharing facilities, we are not limited to use only one Petri net tool. Additionally, students learn importance of model interchange standards and advantages of having XML-based formats like PNML is.
- Since our students are pursuing to obtain degree in computer engineering, they have good skills in software engineering and Web technologies. Accordingly, they realize techniques for extending current Petri net tools with modules for model transformation (e.g. XSLT when we use XML).

Beside these advantages that have been noticed while using P3 in practice, the authors believe that it would make a significant advancement if we:

- Implement support for structuring mechanisms, such as pages and references to the nodes of Petri nets, as well as modules and instances.
- Make component libraries that would model most often used hardware mechanisms such as memory decoders, memory, arithmetical-logic units, DMA, etc. It would be a good illustration of current hardware engineering modeling.
- Implement tools for dynamic model analysis, beside structured tools that are already implemented in the current version of the P3.
- Implement complex tools for model analysis, such as deadlock detection, which would enable students to understand the importance of analysis tools.
- Provide support for Petri net Web-based educational infrastructure in terms of development of both the Petri net ontology (Gašević & Devedžić, 2004) and Petri net Web Service (Havram et al., 2003). In that way we would obtain better Petri net learning features using recent Web-based educational system proposals (Devedžić, 2003) (e.g. create Petri net learning system that will use RDF-annotated Petri net SVG representation in accordance with the Petri net ontology).

Conclusion

Development of P3 software support for studying Petri nets was motivated by the need to teach parts of the Architecture and Organization of Computers course and to employ Petri nets for hardware modeling. In order to give educational character to the P3 software, we have: developed graphical user interface that supports discovering syntax errors in Petri nets and informs users about them; introduced interactive simulation and model analysis; provided possibility for the model exchange with other Petri nets software tools based on PNML. The abbreviated version of the P3, its technical description, as well as developed XSLTs can be downloaded from http://www15.brinkster.com/p3net. Our first experiences in using the P3 tool are quite positive. Especially, evident improvements have been mentioned in the topics that consider parallel and distributed systems (multiprocessors, matrix processors, etc.). From the students' questionnaire we obtained some very useful suggestions for future P3's versions.

We hope that our experiences could be useful for those who teach and use Petri nets in different computer science courses. Especially, we think that P3's interoperability demonstrates valuable improvements of current Petri net software support for teaching. Also, we believe that P3 can be useful for other courses such as Expert Systems, Computer Networks and Communications, and Programming Languages.

Our on-going effort is to develop a Petri net Web-based educational system. This system should consist of a Petri net ontology and a Petri net Web service. Currently, we are extending P3 with support for RDF, as well as for RDF-based annotation of SVG Petri net models. We will use annotated SVG documents in collaboration with the Petri net Web Service. Furthermore, we will use this annotation principle to develop a Petri net Web-based learning environment as well as to make a relation between Petri net models and Learning Object Metadata (LOM) repositories.

References

Billington, J., Christensen, S., van Hee, K., Kindler, E., Kummer, O., Petrucci, L., Post, R., Stehno, C., & Weber, M. (2003). The Petri Net Markup Language: Concepts, technology, and tools. *Paper presented at the 24th International Conference on Applications and Theory of Petri Net*, 23-27 June, Eindhoven, The Netherlands.

Carlson, D. (2001). *Modeling XML applications whit UML: Practical e-business applications,* Boston, USA: Addison-Wesley.

Cassel, L., Kumar, D., Bolding, K., Davies, J., Holliday, M., Impagliazzo, J., Pearson, M., Wolffe, G. S., & Yurcik, W. (2001). Distributed expertise for teaching computer organization & architecture (ITiCSE 2000 Working Group Report). *ACM SIGCSE Bulletin, 33* (2), 111-126.

Chu, Y. (1972). *Computer organization and microprogramming*, New Jersey: Prentice-Hall, Englewood Cliffs.

Denning, P. J. (2000). Computing the profession. In Greening, T. (Ed.) *Computer Science Education in the 21st Century*, New York: Springer-Verlag, 27-46.

Desel, J. (2000). Teaching system modeling, simulation and validation. *Paper presented at the 32nd Conference on Winter simulation*, 10-13 December, Orlando, FL, USA.

Devedžić, V. (2003). Key issues in next-generation Web-based education. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews, 33* (3), 339-349.

Ferraiolo, J. (2001). *Scalable Vector Graphics (SVG) 1.0 Specification - W3C Recommendation*, retrieved October 14, 2004 from http://www.w3.org/TR/2000/REC-SVG-20010904.

Gašević, D., & Devedžić, V. (2003). Software support for teaching Petri nets: P3. In Devedžić, V., Spector, J., Sampson, D. & Kinshuk (Eds.) *Proceedings of 3rd IEEE International Conference on Advanced Learning Technologies*, Los Alamitos, USA: IEEE Computer Society, 300-301.

Gašević, D. (2004). Petri net ontology. *PhD Dissertation*, School of Business Administration, University of Belgrade, Belgrade, Serbia and Montenegro.

Gašević, D., Devedžić, V., &Veselinović, N. (2003). P3 - Petri net educational software tool for hardware teaching. *Paper presented at the 10th Workshop Algorithms and Tools for Petri nets*, 26-27 September, Eichstätt, Germany.

Gilbert, J. E., Hübscher, R., & Puntambekar, S. (2001). Assessment methods in web-based learning environments & adaptive hypermedia - Editors' Introduction. *Paper presented at the AIED Workshop Assessment Methods Web-Based Learning Environments Adaptive Hypermedia*, 19 May, San Antonio, TX, USA.

Havram, M., Gašević, D., & Damjanović, V. (2003). A component-based approach to Petri net Web service realization with usage case study. *Paper presented at the 10th Workshop Algorithms and Tools for Petri nets*, 26-27 September, Eichstätt, Germany.

Jeffrey, J. M. (1991). Using Petri nets to introduce operating system concepts. In Dale, N. (Ed.) *Paper presented at the SIGCSE Technical Symposium on Computer Science Education*, 7-8 March, San Antonio, USA.

Kindler, E., & Weber, M. (2001). The Petri Net Kernel - An infrastructure for building Petri net tools. *Software Tools for Technology Transfer (STTT), 3* (4), 486-497.

Kummer, O., & Wienberg, F. (2000). Renew - the Reference net workshop - tool demonstrations, *Paper presented at the 21st International Conference on Application and Theory of Petri Nets*, 26-30 June, Århus, Denmark.

Kummer, O., Wienberg, R., & Duvigneau, M. (2001). *Renew - XML format guide*, retrieved 12 February 2004 from http://www.renew.de.

Mitrović, A., Martin, B., & Mayo, M. (2002). Using evaluation to shape ITS design: Results and experiences with SQL-tutor. *User Modeling and User-Adapted Interaction, 12* (2), 243-279.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE, 77* (4), 541-580.

Peterson, J. L. (1981). *Petri net theory and the modeling of systems*, New Jersey: Prentice Hall, Englewood Cliffs.

PNML (2003). *PNML home page*, retrieved October 23, 2004, from http://www.informatik.hu-berlin.de/top/pnml.

PNTDB (2003). *Petri Nets Tools Database,* retrieved October 23, 2004, from http://www.daimi.au.dk/PetriNets/tools/db.html.

Reisig, W. (1985). *Petri nets: An introduction*, Berlin: Springer-Verlag.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2003). *Applied operating system concepts*, New York: John Wiley.

Stallings, W. (2000). *Computer organization and architecture*, New Jersey: Prentice Hall.

Štrbac, P. (2002). An approach to modeling communication protocols by using Upgraded Petri nets. *PhD dissertation*, Military academy, Belgrade, Serbia and Montenegro.

Weber, M., & Kindler, E. (2003). The Petri Net Markup Language. *Lecture Notes in Computer Science, 2472*, 124-144.

Wilson, B. G., Jonassen, D. H., & Cole, P. (1993). Cognitive approaches to instructional design. In Piskurich, G. M. (Ed.) *The ASTD handbook of instructional technology*, New York: McGraw-Hill, 21.1-21.22.