Various Kinds of Petri Nets in Simulation and Modelling

Tomáš Vojnar

Department of Computer Science and Engineering, Technical University of Brno
Božetěchova 2, CZ–612 66 Brno, Czech Republic
e-mail: vojnar@dcse.fee.vutbr.cz

Abstract: *This article presents several different kinds of Petri nets, together with computer tools based on them, and compares possibilities of their application in the area of simulation and modelling. Place/transition Petri nets, stochastic Petri nets, hierarchical coloured Petri nets and finally object oriented Petri nets are mentioned here. As for the tools, Design/CPN, TimeNET, LOOPN++ and PNtalk are briefly described and compared. The advantages and disadvantages of object orientation in Petri nets in connection to the system PNtalk are especially emphasized.*

Key words: Petri nets, stochastic Petri nets, coloured Petri nets, hierarchical coloured Petri nets, object oriented Petri nets, simulation and modelling.

1 Introduction

Flexible manufacturing systems, telecommunication and computer networks, real-time systems or information systems are — at least from some points of view — examples of discrete-event systems. All these systems are becoming more and more complex and we can usually characterise them by adjectives like distributed, parallel, safety critical and so on. As their complexity increases so investments to such systems and their design are also shooting up. To design them correctly, to satisfy requirements connected to them and to achieve profit it is necessary to have computer design support tools which allow simulation or even better analysis of behaviour of systems under development.

Petri nets provide formalism which can be very well used in the design process of the systems mentioned above. The behaviour of systems specified by means of Petri nets can be simulated and some of their properties can be checked analytically. Not only properties concerning logical correctness like safeness or liveness are taken into account but after the addition of time also performance aspects can be investigated. Apart from that, Petri nets describe systems in a graphical way which is often better to understand, especially for people who do not work in the area of computers [1].

However, there are many various kinds of Petri nets and computer tools for using them, which sometimes differ quite a lot, as for their expressive power, legibility of models or analytical capabilities. Therefore the aim of this article is to describe briefly and informally some of the Petri net classes, together with tools based on them, and compare their pros and cons in the area of simulation and modelling of descrete-event systems.

The rest of this article is organised as follows: In the first section stochastic Petri nets and the tool TimeNET are described. Then coloured Petri nets and hierarchical coloured Petri nets are mentioned. In the following section two kinds of object oriented Petri nets are discussed, together with the tools LOOPN++ and PNtalk. Finally some concluding remarks are given.

2 Stochastic Petri Nets

Stochastic Petri nets (SPNs) arise when we add stochastic extensions to some pure Petri net formalism, e.g. to place/transition Petri nets (PTPNs) or coloured Petri nets[1]. The classical approach [3], which we will refer to in this section, is to use PTPNs, which offer the easiest ways of analysis. Time extensions of PTPNs can be defined[2] in such a way that every timed transition has a timer, which after having been started with some value (usually derived from a probabilistic distribution) decrements as long as the transition is enabled and the transition fires when the timer reaches zero. Three basic memory policies specifying the way of keeping track of the past are introduced: resampling (transition timers are restarted after every firing), enabling memory (only disabled transition timers are restarted) and age memory (timers are restarted only when they reach zero). In the case of multiple enabling we distinguish between single server and multiple server transition semantics. Several classes of SPN are then distinguished [3, 4] according to the possible time ditributions.

Generalized stochastic Petri nets (GSPNs) allow timed transitions with exponentially distributed delays as well as immediate transitions. Transition priorities are used to avoid confusion and undesirable conflicts. The lowest priority level is reserved for timed transitions. Immediate transition weights say how transitions from extended conflict sets should be fired. GSPNs are a very interesting class of SPNs because every GSPN has an underlying continuous-time Markov process. *Deterministic and stochastic Petri nets* (DSPNs) extend GSPNs in the sense of allowing transitions with deterministic delays. DSPNs do not belong to the class of Markovian SPNs but their steady-state analysis is still possible, provided that at most one deterministically timed transition can be enabled at a time. Therefore DSPNs do not allow more determistic transitions to be concurrently enabled. This is not the case of *concurrent DSPNs*, which can be analysed by means of steady state approximation. Analysis methods have also been proposed for the class of *extended DSPNs* where expolynomially timed transitions can be used with a restriction that at most one non-exponentially timed transition can be enabled at a time. Expolynomial distribution covers many very well known distributions, e.g. uniform or triangular distribution.

As a good example of a tool dealing with SPNs we can use *TimeNET*, which has been developed at the Technical University in Berlin. TimeNET offers analysis methods for GSPNs, DSPNs, extended DSPNs and an approximation method for concurrent DSPNs. Apart from that, simulation of arbitrary SPNs is possible. Both steady-state and transient solutions are supported here. Reward measures are specified by means of a simple language in a specialised editor. Another specialised editor is used also for defining expolynomial distributions. As for the simulation, both interactive token game and fast simulation are included. In the case of fast simulation, initial transient phase detection and automatic run length detection are embedded in the tool. Multiple experiments are possible too.

An example of an SPN, which was drawn and simulated in TimeNET, is shown in figure 1. It is a simple model of a small medical centre. So that it will be possible to analyse this net, it is necessary to make its state space finite, e.g. by adding a new place with finite initial marking as an input place of transition `Entrance` and output place of transitions `LeaveWithoutExp` and `Expert`. It is also necessary to replace at least one expolynomially timed transition by an

[1] The term SPN is reserved for nets with only exponentially timed transitions in [3] but will use it in a more general way, how it is common e.g. in [4].

[2] It is of course possible to define time extensions in many other ways. Some of them are discussed in the following sections with respect to the high-level Petri nets.

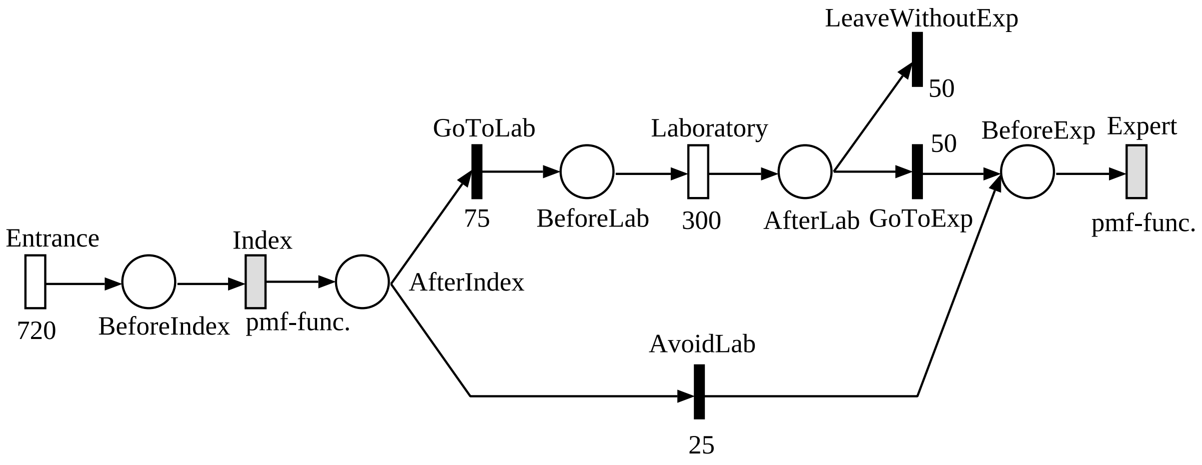

Figure 1: A simple medical centre described by an SPN

exponentially timed one because of the possibility of their concurrent enabling. Results obtained by means of simulation of the model from figure 1 in TimeNET are presented in table 1 together with the specification of corresponding reward measures.

Result Par.	**Definition**	**Value**
IndQueueLength	E{#BeforeInd}	0.147
IndUse	P{#BeforeInd > 0}	0.135
LabQueueLength	E{#BeforeLab}	0.477
LabUse	P{#BeforeLab > 0}	0.335
ExpQueueLength	E{#BeforeExp}	2.792
ExpUse	P{#BeforeExp > 0}	0.856

Table 1: Reward measures and corresponding simulation results for the net from figure 1

We can conclude that described SPNs have a very attractive feature in the possibility of their exact logical as well as performance analysis. Unfortunatelly, this analysis is often very time and space consuming and cannot be used in the case of large models. It is also limiting that we usually have to deal with finite state spaces only. Nevertheless, the main drawback of SPNs based on PTPNs is the fact that large models are very complicated and badly legible. Therefore colours and some structuring mechanism are to be used.

Speaking about TimeNET, we can say it is a very good tool, but of course it could be improved as well. For example it is difficult to find out the maximum possible number of tokens in a place or the time which tokens on average need to proceed from one place to another distant one. Also the user interface could be improved by incorporating some object oriented techniques into the work with graphical objects.

3 Coloured Petri Nets

Coloured Petri nets (CPNs) extend basic PTPNs by adding colours (i.e. data elements) to tokens and using expressions working with them. There are many various definitions of CPNs but Kurt Jensen's definition [1] is probably the best known. In this definition arc expressions specify tokens which are being added or removed by transitions. Initial marking is determined by initialization functions. Both places and variables of expressions are typed. A functional ML-like language is used for expressing data types as well as expressions.

It is obvious that another definitions of CPNs can use different inscription languages, they can join expressions with transitions instead of arcs, they can omit types and so on. However, these definitions are beyond the scope of this article.

Introduction of colours significantly reduces sizes of models thus increasing their legibility. Nevertheless, this is still not enough and therefore various structuring techniques have been proposed. Substitutions of transitions, fusions and substitutions of places and transition invocations are the most common ones. The first two of them are used in Kurt Jensen's definition of *hierarchical CPNs* (HCPNs).

So that it will be possible to explore performance characteristics of models, it is necessary to introduce time into CPNs. This can be done for example by adding time stamps [2] to tokens. These time stamps are determined by the current model time, when tokens are created, which can be modified by transition and arc time expressions. Adding positive amount of time by output arcs or transitions models duration of operations while interruption of these operations can be modelled by positive time amount on input arcs.

One of the best tools based on HCPNs is *Design/CPN*, which has been developed at Aarhus University. It can deal with very large industrial models, which was proved by modelling several industrial and army systems (VLSI chip, BRI protocol for ISDN networks, radar surveillance system, ...). Design/CPN offers simulation as well as analysis. During simulation code segments written in SML can be used to produce statistics, to visualise processes running in the model etc. As for the analysis, it is currently possible to construct occurrence graphs and use them for proving model properties. This is done by means of SML functions working over occurence graphs. These functions are predefined for several well known logical properties like liveness, home properties and others. Besides, the user can add his own functions. Simulation and described analysis can be performed even for timed nets. Moreover, special kind of temporal logic — ASK-CTL — is defined over occurrence graphs and can also be used for properties proving [5].

Of course, HCPNs and Design/CPN have several weaker features as well. Transition invocation is not supported and thus recursive definitions cannot be used. Further, it is difficult to create a subnet and communicate with it from several places of the corresponding super-page[3]. It has also turned out that using colours and expressions can reduce the model but, on the other hand, dealing with inscription language can be difficult for people who do not know it and especially for those who do not work in the area of computers. This problem can be solved by introducing dedicated HCPNs which use only restricted number of token colours and expressions and these can be defined by means of special graphical symbols. Dedicated HCPNs have been introduced in [6, 7] and exploited for flexible manufacturing systems modelling. An example of such a net is shown in figure 2. Another approach to reduce the complexity of net inscriptions is to use a very simple untyped language, which is the case of PNtalk described in the next section.

Design/CPN itself does not have a very good support for collecting timed statistical information, i.e. it is difficult to weigh values by the time they are valid for. Performance analysis in the form of transformation to Markovian processes is not supported here and neither are multiple experiments. The tool can deal entirely only with finite occurence graphs. It is not easy to specify reward measures — they must be built in the model. Finally, there are problems with performing actions after the end of simulation, which can be useful e.g. when working with statistics.

[3]This problem has been successfully solved in object oriented Petri nets by encapsulating nets into objects referred from several places and accessed by message sending.

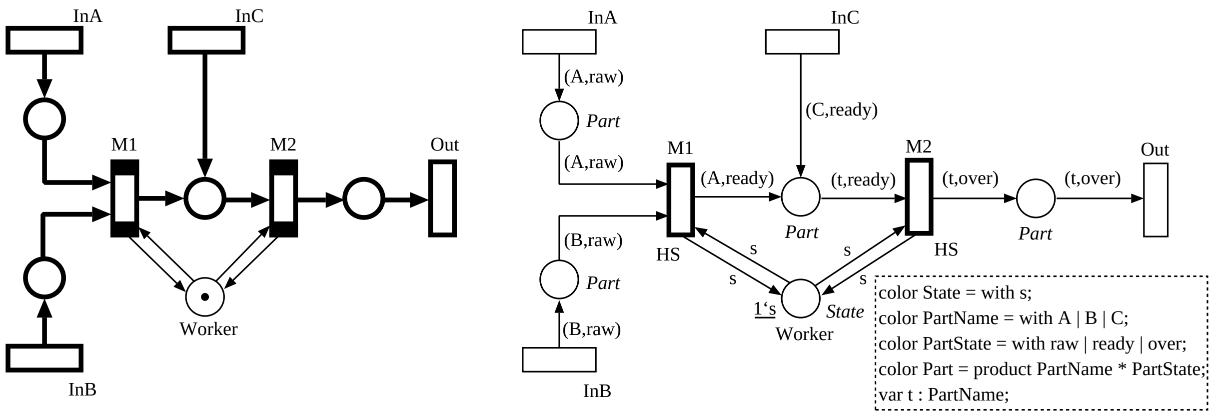

Figure 2: A simple manufacturing system modelled by classical and dedicated HCPNs

4 Object Oriented Petri Nets

Object oriented Petri nets are based on CPNs which they enrich by object oriented structuring paradigm, i.e. there appear classes, objects, methods and so on. This step makes it possible to encapsulate some properties and behaviour into objects, which can be bound to tokens and referred from several places of the model. Introducing methods is similar to transition invocation and it allows the writing of recursive definitions. Inheritance offers a possibility of incremental specification of classes and nets bound to them.

However, it is still advantageous to support also classical structuring techniques, which can be very well used in some situations. Substitution of transitions provides an easy way for creating channels between nets because port and socket places are joined and thus tokens flow straight from one net to another. If we do not have this possibility, we have to use two separate objects with references to each other and transfer tokens by calling appropriate methods, which can be sometimes a bit tedious. Similarly, it is a good idea to support fusion of places, which can be used e.g. for avoiding crossing of too many arcs.

There are again several definitions of OOPNs. Let us first have a quick look at the OOPNs connected to the language *LOOPN++* [8], which has been developed at the University of Tasmania. This language supports classes which consist of fields, functions and actions. Fields are components which hold data elements and can be used as Petri net places. If we use fields of appropriate object type with redefined input and output actions, they can play the role of substitution places or places with capacity. Actions have synchronised input, output and call parts and they can constitute transitions. Functions are parameterised expressions, which are not described by means of Petri nets. Objects in LOOPN++ can communicate by message sending (which do not change state of objects), places sharing (which is similar to substitution of transitions) or by data passing to and from object boundaries (i.e. by adding and removing data elements to and from an impilicit field “self”).

There is a compiler from LOOPN++ to C++ which can produce programs usable as running prototypes of modelled systems. However, it is necessary to describe Petri nets in a textual form (a graphical tool is currently under development) and this is rather unpleasant.

Another class of OOPNs is joined with the language *PNtalk* [9, 10] and it has been developed at the Technical University in Brno. PNtalk uses Smalltalk as its inscription language. Smalltalk is purely object oriented and, what is more, it is untyped and rather simple. Therefore net inscriptions are also quite simple. In PNtalk not only objects’ own behaviour is described by Petri nets but also methods are. Non-atomic methods as well as atomic predicates can be defined over classes here. Actions in PNtalk are not connected to arcs as in Design/CPN but to transitions. Recently substitutions of transitions, fusions of places and time have been added to PNtalk. Time is defined in such a way that timed transitions remove input tokens and add output tokens after some delay (possibly derived from a probabilistic distribution).

PNtalk has a graphical integrated environment, in which OOPNs can be defined and simulated. Classes are being displayed in so called browser, which is typical also for Smalltalk. Particular methods are being displayed separately, which allow the modeller not to deal with too big nets. Both interactive and fast simulation are supported. There is a possibility to use all classes from Smalltalk in PNtalk, which makes it possible to realise easily statistics, graphical presentations and so on.

Unfortunatelly, PNtalk is in the early stage of its development and therefore it is not very efficient and cannot deal with large nets. It also does not have any support for classical logical analysis neither for performance analysis yet. What is perhaps worse is the fact that the way in which time is incorporated into PNtalk does not allow interruption of timed actions once they are started.

At the end of this section we should emphasise that if we have a modelling tool based on OOPNs we do not have to use its object oriented features in all possible situations. Thus we can simply work with HCPNs, CPNs or even PTPNs if it is convenient. We can guess that object oriented features will not bring significant gain before we have to deal with huge models. Tiny systems can sometimes be modelled more conveniently by simpler classes of Petri nets!

According to what is said in the previous paragraph, there are, for example, two possible approaches to modelling shared resources in PNtalk (and other systems based on OOPNs). Firstly, we can model them classically by shared places (see figure 1). Secondly, we can define special classes representing these resources. Then we will be able to seize and release such resources by sending them appropriate messages. This approach removes many arcs going to and from shared places and makes the model more legible. Further, classes modelling shared resources can have built-in statistics offering an easy way of collecting information about average length of the queue, waiting time etc.

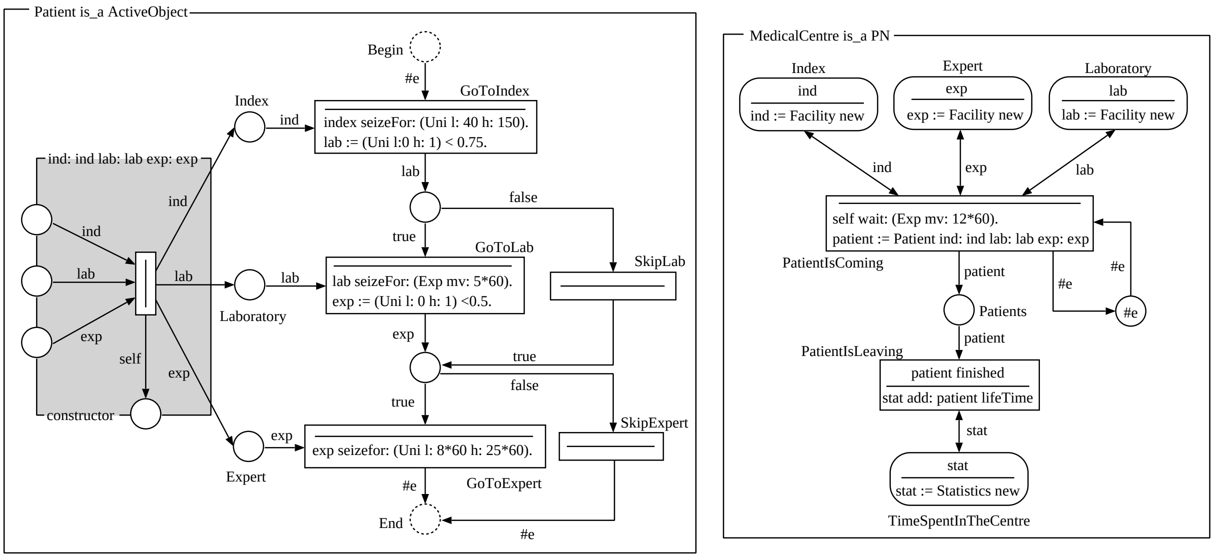

Figure 3: A simple medical centre described by an OOPN

An example of an above described approach is depicted in figure 3. In this example class `ActiveObject` stores automatically times of the beginning and end of its objects' active life and allows testing of their activity. Class `Facility` represents a shared resource. Both of these classes once created can be used in many future models, together with classes representing statistics and random generators.

5 Conclusions

We have shown that there are many various kinds of Petri nets. Object oriented Petri nets are probably the most sophisticated of them all. But, there are situations for which OOPNs are too strong and produce unnecessarily complex models. Similarly, there are cases when object oriented structuring techniques are not the best ones. Therefore OOPNs and tools based on them should also contain features of HCPNs, CPNs and PTPNs and let the modellers choose what they want to use. To allow performance analysis of models it is essential to incorporate time into Petri nets. This should be done in such a way so that not only performing timed actions will be possible but also interrupting them ought to be allowed.

Not surprisingly, even very good Petri net formalism is not worth much if there are no computer tools based on it. What is more, these tools should be efficient enough for dealing with large industrial models. Classical logical analysis as well as performance analysis should be supported. This is due to the fact that the possibility of analysis distinguishes Petri nets from other modelling methods in probably the most significant way despite its time and space requirements — graphical representation, possibility of simulation, hierarchy or object oriented features are being offerred by other techniques too.

At the very end we can say that according to the first concluding paragraph especially PNtalk is in a very good position but TimeNET and Design/CPN are far better when also taking into account the second part of our final remarks.

This work was supported by the grant of the Czech Grant Agency GAČR No. 102/96/0986 and by the Universities Development Fund of the Czech Ministry for Education under the contract F1: 618/1997. A conscise version of this article was published in the proceedings of MOSIS'97.

References

[1] Jensen, K.: *Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use, Volume 1*, Springer-Verlag, 1992.

[2] Jensen, K.: *Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use, Volume 2*, Springer-Verlag, 1995.

[3] Marsan, M.A., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: *Modelling with Generalized Stochastic Petri Nets*, Wiley Series in Parallel Computing, 1995.

[4] German, R., Kelling, C., Zimmermann, A., Hommel, G.: *TimeNET — A Toolkit for Evaluating Non-Markovian Stochatic Petri Nets*, Performance Evaluation 24, 1995, pp 69–87.

[5] Cheng, A., Christensen, S., Mortensen, K.H.: *Model Checking Coloured Petri Nets Exploiting Strongly Connected Components*, Technical Report, University of Aarhus, 1996.

[6] Zimmermann, A.: *A Modelling Method for Flexible Manufacturing Systems based on Colored Petri Nets*, Proceedings of International Workshop on New Directions of Control and Manufacturing, Hong-Kong, 1994, pp. 147–154.

[7] Zimermann, A., Bode, S., Hommel, G.: *Performance and Dependability Evaluation of Manufacturing Systems Using Petri Nets*, Proceedings of 1st Workshop on Manufacturing Systems and Petri Nets, 17th International Conference on Application and Theory of Petri Nets, Osaka, 1996, pp. 235–250.

[8] Lakos, C.A., Keen, C.D.: *LOOPN++: A New Language for Object-Oriented Petri Nets*, Technical Report R94-4, University of Tasmania, 1994.

[9] Janoušek, V.: *Objektově orientované Petriho sítě — formální základ jazyka PNtalk*, Technická zpráva, ÚIVT FEI VUT v Brně, 1997.

[10] Vojnar, T.: *Systém PNtalk*, Technická zpráva, ÚIVT FEI VUT v Brně, 1997.