# Petri Net Modelling and Performability Evaluation with TimeNET 3.0

Armin Zimmermann, Jörn Freiheit, Reinhard German, and Günter Hommel

Performance Evaluation Group, Technische Universität Berlin, FR 2-2,
Franklinstr. 28/29, 10587 Berlin, Germany,
{azi | freiheit | rge | hommel }@cs.tu-berlin.de

**Abstract.** This paper presents TimeNET, a software tool for the modelling and performability evaluation using stochastic Petri nets. The tool has been designed especially for models with non-exponentially distributed firing delays. A general overview of the software package and its new features is given. The graphical user interface is completely rewritten. It integrates different model classes in a user-friendly and consistent way. One of the recent enhancements is an environment for the modelling and performance evaluation of manufacturing systems based on coloured stochastic Petri nets. A manufacturing system is modelled and analysed as an application example.

## 1 Introduction

Model based performance evaluation of technical systems is a powerful and inexpensive way of predicting the performance before the actual implementation. Its importance increases with the complexity of the applications. Discrete event systems like computers, communication systems, and manufacturing systems are well-known examples. Failures and repairs make their behaviour even more unpredictable. The impact of the dependability on the system performance is captured in the notion of *performability* [18].

Petri nets are a graphical method for the convenient specification of discrete event systems. They are especially useful for systems with concurrent, synchronised, and conflicting activities. Evaluation of the performance is facilitated by associating stochastic firing delays with transitions. Basic quantitative measures like the throughput, loss probabilities, utilisation and others can be computed.

Since the modelling framework of stochastic Petri nets has been proposed (overview of formalism in [1]), many algorithms and their implementations as software tools have been developed (e.g. GreatSPN [3], SPNP [5], UltraSAN [20], HiQPN [2], DSPNexpress [17] [16]). Most tools provide a graphical user interface for the convenient drawing and editing of the model. They usually include analysis components for one or several classes of models. There are two main types of performance evaluation algorithms: direct numerical analysis or discrete-event simulation. Both techniques can be used to compute performance measures either in steady-state or after a transient time.

TimeNET [10] is partially based on an earlier implementation of DSPNexpress [16] [17], also performed at the Technical University Berlin [15]. TimeNET is intended for the evaluation of non-Markovian SPNs. Under the restriction that all transitions with non-exponentially distributed firing times are mutually exclusive, stationary numerical analysis is possible [6] [11]. If the non-exponentially timed transitions are restricted to have deterministic firing times, transient numerical analysis is also provided [8] [12]. In case of concurrently enabled deterministically timed transitions, an approximation component based on a generalized phase type distribution is offered as well [9].

If the mentioned restrictions are violated or the reachability graph is too complex for a model, an efficient simulation component is available [14]. A master/slave concept with parallel replications and techniques for monitoring the statistical accuracy as well as reducing the simulation length in the case of rare events are applied [13]. Analysis, approximation, and simulation can be performed for the same model classes. TimeNET therefore provides a unified framework for modelling and performance evaluation of non-Markovian stochastic Petri nets.

Recent enhancements of TimeNET include a component for the steady-state and transient analysis of discrete time deterministic and stochastic Petri nets [7] [22]; a component for the modelling, steady-state analysis, and simulation of stochastic coloured Petri nets, especially for the performance evaluation of manufacturing systems [23] [27] [25]; and a completely rewritten graphical user interface, which integrates all different net classes and analysis algorithms. Moreover, using a coloured Petri net model of a manufacturing system, the actual system can be controlled directly by TimeNET.

This paper describes the new version of the tool TimeNET. It has been successfully applied during several modelling and performance evaluation projects.

This paper is organised as follows. Section 2 contains a description of the supported net classes and analysis methods of TimeNET. The usage of the tool and information related to the new interface of TimeNET are given in section 3. Modelling and performance evaluation of a manufacturing system is illustrated in section 4 using an application example. Finally, concluding remarks are given.

## 2 Supported Net Classes and Analysis Methods

The software package TimeNET offers two different modelling and evaluation environments. Markov regenerative stochastic Petri nets (MRSPNs) with uncoloured tokens and their corresponding evaluation techniques are described in subsection 2.1. The part of the tool dealing with coloured Petri nets is presented in subsection 2.2. Coloured models can contain the same stochastic extensions like MRSPNs.

### 2.1 Markov Regenerative Stochastic Petri Nets

The term "Markov regenerative stochastic Petri nets" has been introduced in [4] and is now used in literature. In [11] [10] the term "extended deterministic and

stochastic Petri nets (eDSPNs)" has been used for a similar model class. However, this model class is closely related to the usual class of Markovian stochastic Petri nets, popular examples are generalized stochastic Petri nets (GSPNs) [1] or stochastic reward nets (SRNs) [19]. It consists of places and transitions, which are connected by input, output, and inhibitor arcs. Places are drawn as circles and can contain indistinguishable tokens, while transitions are depicted as rectangles. Convenient modeling elements known from SRNs like guards and marking-dependent arc cardinalities are also allowed. The main distinctive feature of the framework is the with respect to the timed transitions: the firing times may also be deterministic (referred to as "deterministic transitions", drawn as filled rectangles) and generally distributed (referred to as "general transitions", drawn as shaded rectangles). The general distribution may be one of the class of so-called "expolynomial" ones, which may be represented piecewise by polynomials and expolynomials. This class contains common distributions like the uniform or the triangular one. The firing semantics of the general transitions can either be "race enabling policy" or "age memory policy" (in more recent papers also referred to as "preemptive repeat different" and "preemptive resume").

As usual, the current state of the model is given by the vector of token numbers in all places and is referred to as the marking. The reachability graph is defined by the set of vertices corresponding to the markings reachable from the initial marking and the edges corresponding to transition firings. If an immediate transition is enabled in a marking, no time is spent in it during the marking evolution. The reachable markings can be partitioned in vanishing and tangible markings accordingly.

The behaviour of the model is given by the initial marking and the subsequent transition firings, describing a stochastic process [6]. The type of process depends on the types of transitions and certain enabling conditions. The firing delay of transitions in TimeNET can either be zero (immediate), exponentially distributed, deterministic, or belong to a class of general distributions called *expolynomial*. Such a distribution function can be piecewise defined by exponential polynomials and has finite support. It can even contain jumps, making it possible to mix discrete and continuous components. Many known distributions (uniform, triangular, truncated exponential, finite discrete) belong to this class.

The analysis in the Markovian case is performed as usually. In the MRSPN case it is required that the general transitions are mutually exclusive (i.e., two of them may not be enabled in the same marking) and have firing semantics preemptive repeat different. The stationary analysis is then based on the construction of an embedded Markov chain [6]. If the MRSPN is restricted to a DSPN (the general transitions are deterministic), the transient analysis is also possible. It is based on the method of supplementary variables [11] [12].

TimeNET also provides an approximation component for DSPNs with concurrently enabled deterministic transitions based on generalized phase type expansion [9]. During the transient analysis, TimeNET shows the evolution of the performance measures from the initial marking up to the transient time graphically.

TimeNET also comprises a simulation component [14] [13], which evaluates eDSPN models without the enabling restriction. The simulation component of TimeNET can be applied for the steady-state and transient evaluation.

A recent addition to TimeNET are components for the analysis of models with a discrete time scale [7] [22], allowing for geometric distributions, deterministic times and discrete phase type distributions. This model class is referred to as *discrete deterministic and stochastic Petri nets* (DDSPN). Due to the mapping technique on a discrete time Markov chain there is no enabling restriction.

### 2.2 Coloured Petri Nets for Manufacturing Systems

Manufacturing systems are a classical application area of Petri nets; see [21] for a recent survey. The class of coloured stochastic Petri nets used by TimeNET has been introduced especially for the modelling of manufacturing systems, although it is not restricted to it. Two colour types are predefined: *Object tokens* model work pieces inside the manufacturing system, and consist of a name and the current state. *Elementary tokens* cannot be distinguished, and are thus equivalent to tokens from uncoloured Petri nets. Places can contain only tokens of one type. Textual descriptions needed in coloured Petri nets for the definition of variables and colour types can be omitted, and the specification of the types of places and arcs are implicitly given. TimeNET supplies a library of template models for typical machines and their failure-and-repair behaviour. The models are hierarchically structured, which is necessary to handle complex systems.

Structure and work plans are modelled independently using this net class. This is important for the evaluation of different production plans, where the structural model is not changed. The structural model describes the abilities and work plan independent properties of the manufacturing system resources, such as machines, buffer capacities, and transport connections. Production sequence models specify the work plan dependent features of the manufacturing system. Each route can be thought of as a path through the manufacturing system. Later on, the different model parts are automatically merged resulting in a complete model, which then includes both the resource constraints of the system and the synchronisation of the production steps.

After specifying a manufacturing systems model using the graphical interface, its performance and dependability can be evaluated. Before doing so, one should check whether the model correctly describes the system. Structural analysis methods can be used for this task, which are currently being implemented. A second debugging tool is the token game, showing the states and actions of the system. With TimeNET the modeller is able to either interactively check the model behaviour or to watch an automatic animation of it. However, the main focus is on efficient quantitative evaluation of performance and dependability measures.

Firing delays are associated with transitions for the performance evaluation. We adopt the set of distributions as defined for eDSPNs here. They include immediate, exponential, deterministic, and more general transitions. TimeNET provides components for the steady-state analysis and simulation. The numerical

analysis techniques for eDSPNs as described above have been adapted to the coloured net class. This is possible because the stochastic process belongs to the same class.

Details of the component for manufacturing system modelling and evaluation can be found in [23] [27] [25]. There are two additional components which are outside the scope of this paper and are therefore only briefly mentioned. Users without knowledge of Petri nets can model a manufacturing system with functional blocks originating from the field of production management [26]. Those models are automatically translated into coloured Petri nets and analysed. The provided performance evaluation algorithms can therefore be used without the need to use Petri nets. A recent enhancement is the possibility to directly control a manufacturing system with the token game or animation feature of TimeNET.

## 3 Agnes – The New TimeNET Interface

A powerful and easy to use graphical interface is an important requirement during the process of modelling and evaluating a system. For version 3.0 of TimeNET a new generic graphical user interface has been developed. It is called "agnes" (**a** **g**eneric **n**et **e**diting **s**ystem).

### 3.1 Tool Usage

All currently available and future extensions of net classes and their corresponding analysis algorithms are integrated with the same "look-and-feel" for the user. Figure 1 shows a sample screen shot of the interface during a modelling session with coloured Petri nets.

The upper row of the window contains some menus with basic commands for file handling, editing, and adjusting display options. Under the menu item "Module" the analysis algorithms applicable for coloured Petri nets can be accessed. Some of the most frequently used functions are also available by clicking one of the icons below. The current mouse position and zoom level is shown on the right. Below the icon list the current net class is displayed (HCPN for hierarchically coloured Petri net in this case). The net objects are displayed on buttons in the next row. Each one is shown as an icon and a textual identifier. A group of objects is accessed via each one of these buttons. For instance, all transitions are organised in one group. Clicking on the transition button switches to the next type, but the desired one can also be selected directly by a menu that pulls down from the textual object description.

There are two types of objects available in general, which form the nodes and arcs of the model class. One node object and one arc object is always selected and highlighted. Textual objects like parameter definitions, performance measures, and comments are special cases of node objects without the possibility to connect arcs with them.

The main drawing area contains a part of the current model. It can be edited with the left mouse button like using a standard drawing tool with operations

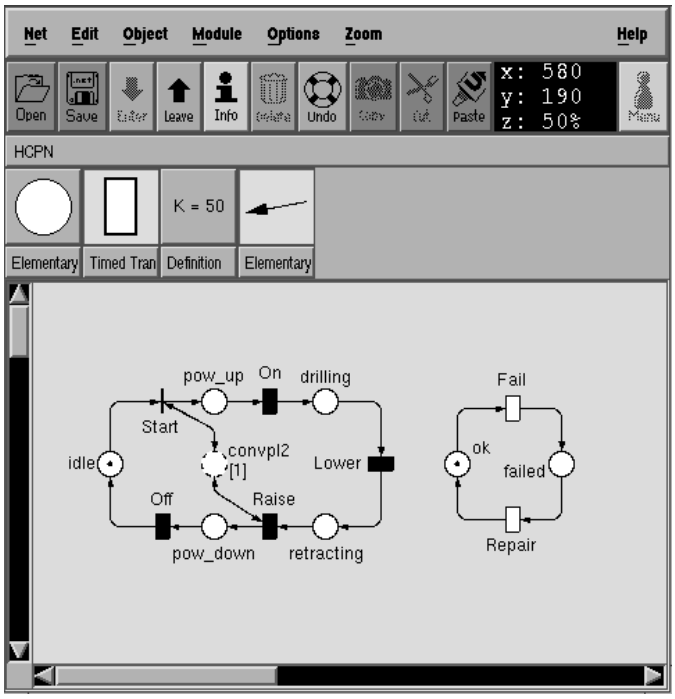


**Fig. 1.** Sample screenshot of the graphical user interface

for selecting, moving, and others. Clicking the middle mouse button creates an object of the currently highlighted type. Dragging from one object to another creates a new arc between them. Double-clicking on any object displays a window with its attribute identifiers and their current values, e.g. for a place the initial marking and the name. If the object is hierarchically refined (like a substitution transition), double-clicking it displays the refining model.

Depending on the net class, different objects are available in the lower icon list. In addition to that, analysis methods typically are applicable for one net class only. Those methods are integrated in the menu structure of the tool, which therefore changes automatically if a different net class is opened (see subsection 3.2).

For eDSPNs there are several additional options. The menu "Validate" contains algorithms for checking the net for syntax errors, computing structural properties like place invariants and extended conflict sets, and the interactive token game. The token game can now be run automatically as an animation of the system behaviour or with the user selecting the firing transition.

Performance evaluation algorithms are accessible in the menu "Evaluation". Methods for the stationary analysis, approximation, and simulation, as well as transient analysis and simulation are available. When an evaluation method is

chosen, the user has to decide whether the underlying time scale should be discrete or continuous. Afterwards, the evaluation can be started or stopped (if running). Several algorithm-dependent options can be set. Examples are the method of solving the linear system of balance equations and the confidence interval of the simulation.

It is often necessary to evaluate the system performance for different parameter values in some given range. The corresponding "experiment" feature of TimeNET has been improved in the current version. It is now possible to either linearly or logarithmically change the varying parameter. Experiment descriptions can be stored in a file for later reuse. After the tool has finished all required evaluations for the parameter range, the results are automatically plotted using *gnuplot*.

During the transient analysis and simulation, a window displays the transient development of the specified performance measures. A graphical editor for specifying probability density functions of transitions with generally distributed firing delays is available.

If a manufacturing system is modelled and evaluated with TimeNET 3.0, several different net classes with their own applicable algorithms are used. The modeller can start with function symbols and translate them into a Petri net with separate models of structure and processing sequences. For this task as well as the modelling of large manufacturing systems with e.g. similar machines the library of Petri net submodels is used. The library in itself is also edited with its own net class in TimeNET. Instantiation of a library model and the translation of function symbol models into Petri nets as well as the compilation of a complete model from the two separate parts are all implemented as net class dependent extensions of TimeNET.

### 3.2 Net Classes and Modules

Software development for TimeNET is usually done by students as part of their diploma or PhD thesises. A major problem is therefore to keep all analysis components modular with well-defined interfaces. Especially the graphical user interface has to be able of integrating different net classes and algorithms and should be easily extendable. The existing interface of the former version of TimeNET was designed for uncoloured Petri nets without hierarchies. Therefore it has been completely rewritten for version 3.0.

A generic interface was implemented, that can be used for graph-like models with different types of nodes and arcs. Nodes can be hierarchically refined by corresponding submodels. The new interface agnes is not restricted to Petri nets. In fact, it is already being used for other tools than TimeNET. It is implemented in C++ and uses the Motif toolkit.

Two design concepts have been included in the interface to make it applicable for different model classes: A *net class* corresponds to a model type and is defined by a text file. In this file, for each node and arc type of the model the corresponding attributes and the graphical appearance is defined. The shape of each node and arc is defined using a set of primitives (e.g. polyline, ellipse, and

text). Shapes can depend on the attribute value of an object, making it possible to show tokens as dots inside places. Figure 1 shows a sample appearance. Nodes can correspond to submodels of a different net class. This facilitates e.g. the organisation of the library submodels like in a file manager.

Functions beyond drawing a model depend on the net class and require programming. Agnes offers the possibility to implement *modules* that are compiled and linked to the program. A module has a predefined interface to the main program. It can select its applicable net classes and extend the menu structure by adding new algorithms. An example of an implemented module applicable without restrictions is an export filter to *xfig*.

With the described user interface, two modellers are able to work together on the same model over the internet. It is possible to start an additional model editor window on a remote host. This window shows the same model as the original one and allows collaborative work on it. Model parts that are currently changed by one modeller are locked for the other.

## 4 A Manufacturing System Application Example

This section demonstrates the use of TimeNET 3.0 for the modelling and performability evaluation of a simple manufacturing system. A real-life industrial application example is considered in [24]. The example chosen here is a manufacturing cell built of parts from the "Fischertechnik" construction kit. It is used for education and research at the department. Figure 2 shows its layout.

New parts are initially stored in the high bay racking on pallets. The rack conveyor can fetch one of them and deliver it to the lower pallet exchange place. A horizontal crane then takes it to the first conveyor belt. The system of three conveyor belts moves the part to one processing station after another. There are two drilling stations, the second having three different interchangeable drilling tools. The last station is a milling machine. Parts stay on the conveyor during processing. After leaving the machines, parts are transported on a turn table. This table puts them into position for the slewing picker arm who takes the part to the upper pallet exchange place. From there it is brought back to a place in the high bay racking by the rack conveyor.

The exchange of new and finished parts takes place via the rack storage. We assume that there are two different types of work pieces to be processed, named A and B. The first one has to be machined by the two drilling machines and the milling machine in this order. An additional drilling and milling operation is required for parts of the second type. The parts are moving counterclockwise through the system.

### 4.1 Structural Model

The described application example is modelled with the class of dedicated coloured Petri nets as described in Section 2.2. A strict separation between the model of

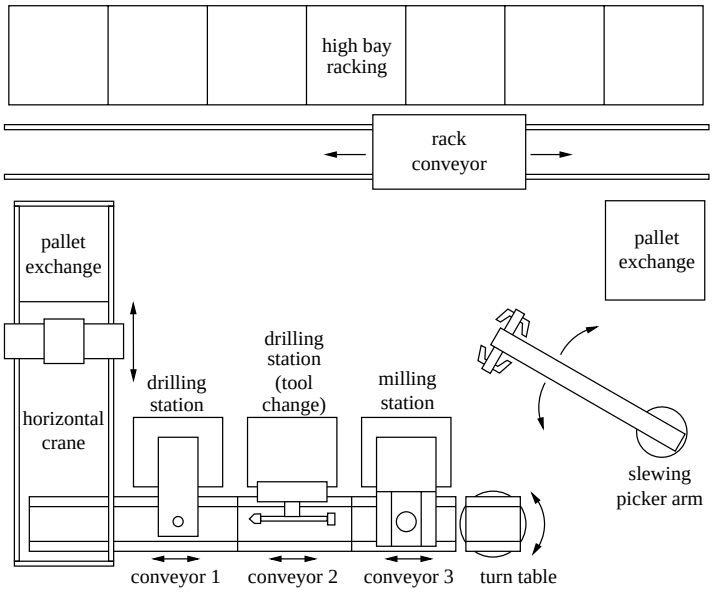


**Fig. 2.** Overview of the modelled system

the manufacturing system's structure and the sequence of the processing steps for each product is observed.

Figure 3 shows the top level of the structural model. Its composition follows the layout of the modelled system, which makes it easier to understand. Places model buffers and other possible locations of parts. The place `rack` corresponds to the rack storage, the places `exchpl1` and `exchpl2` to the pallet exchange places, and place `turnpl` to the turn table. The remaining four places represent the locations of work pieces on the conveyors which are directly in front of the machines or the horizontal crane. As described above, in- and output of parts takes place through the rack storage and is modelled with transitions `input` and `output`.

In principal, there are two different operations that can be performed: transport and processing of work pieces. The former corresponds to moving a token to another place, while the latter is modelled by a change in the colour of the token that corresponds to the work piece. Transitions modelling machines specify processing steps which only change the token colour. This is emulated by removing the former token form the place and instantly adding a token with the new colour during the firing of the transition. Therefore many transitions and places are connected by arcs in both directions, which are conveniently drawn on top of each other. The structural model contains all possible actions of the resources, even if they are not used for the processing. The horizontal crane could e.g. move parts to the left as well.

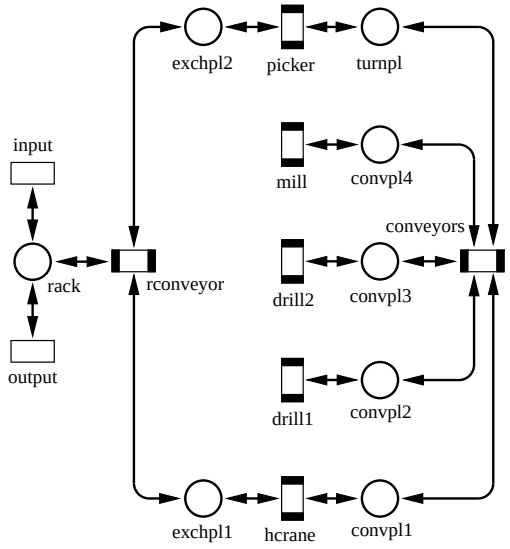


**Fig. 3.** Highest level of the hierarchical coloured model

Transitions with thick bars depict substitution transitions, which are refined by a submodel on a lower level of hierarchy. These transitions are e.g. used to describe the behaviour of a machine with more detail during a top-down design. Submodels from a library of standardised building blocks (templates) can be parameterised and instantiated while refining the model. This alleviates the creation of complex manufacturing system models, where many structurally similar parts can be found.

Transition `rconveyor` contains the model of the high bay rack conveyor, while transitions `hcrane` and `picker` correspond to the horizontal crane and slewing picker arm. For the transport of a part from one machine to the next, two of the three conveyor belts have to operate simultaneously. All three conveyors are therefore treated together as one transport means and modelled by transition `conveyors`. Thus, their synchronisation is hidden at a lower level and can be specified together.

Figure 1 shows the graphical user interface of the tool while editing such a refining submodel. The whole model consists of 11 submodel. The drawing area contains the model part corresponding to transition `drill1` of the high-level model. There is only one interface with the surroundings, place `convpl2`. This place is connected to transition `drill1` at the higher level of hierarchy, and is therefore known at the lower level. As it is only a reference to the real place here, it is drawn as a dotted circle. Place `convpl2` is the only actual buffer of parts in this submodel. All other places correspond to different states of the machine. Therefore they are modelled using elementary places, which are drawn thin and can contain only uncoloured tokens.

The model has two main parts. The right hand part describes the failure and repair behaviour of the drilling machine. Exponentially distributed firing delays are associated with transitions `Fail` (mean time to failure) and `Repair` (mean time to repair). On the left is the model of the detailed steps during one drilling operation. The immediate transition `Start` is enabled if there is a token in place `convpl2` with a colour indicating that the next processing step is a drilling operation at this machine. It fires and therefore enables a firing sequence of transitions `On`, `Lower`, `Raise`, and `Off`. They correspond to powering up, lowering, raising and finally switching off the drilling tool. Marking-dependent firing guards of these transitions ensure that they are not enabled if the drilling machine is failed (no token in place `ok`). The actual processing task is finished and the colour of the token corresponding to the work piece is changed with the firing of transition `Raise`.

A submodel as the shown one can be exchanged for a model e.g. with a different failure behaviour without changing the model at the upper level of hierarchy.

### 4.2 Production Sequence Model

In addition to the structural model for each product a model of the production steps has to be defined. With TimeNET 3.0, the class of coloured Petri nets is used for this task as well. Figure 4 shows the first part of the production sequence model of one work piece.

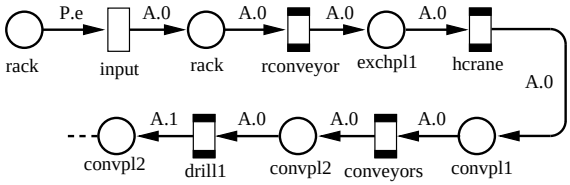


**Fig. 4.** Part of the production sequence model

This model describes the sequence of operations and transports for a part named A at the highest level of hierarchy. Each step can only be carried out by a resource that is available in the manufacturing system layout. Therefore, only transitions, places and their connecting arcs from the structural model can be used here. Arc inscriptions show the name (`A`) and processing state (`0` or `1`) of the work piece, separated by a dot.

The model shown in Figure 4 corresponds to the structural model in Figure 3. Model elements in production sequence models refer to their structural counterpart through the use of identical names. It is obvious that a substitution transition in a production sequence model has to be refined with a submodel.

This submodel is then associated to the submodel of the corresponding substitution transition in the structural model. This relationship between both model parts holds for all submodels in the hierarchy. We use the term *associated Petri nets* for this concept of specifying different views of a system in related model parts.

It is possible to model alternatives in production sequences of work pieces. Logical expressions depending on the current marking as well as probabilities can be used to choose a path for a token at such an alternative. A production sequence model usually consists of a simple succession of transitions and places. An exception is the modelling of (dis-)assembly operations. More than one input or output arc is connected to a transition in this case. Although it cannot be immediately seen in Figure 4, there is also an assembly operation needed for the example production sequence. Each part is transported and processed while being fixed to one pallet. For an input of a new part into the rack storage, there has to be an empty pallet in it (place `rack` contains a token of colour `P.e`). The input operation (transition `input` fires) removes this token and puts back a token with colour `A.0`. The inverse operation is carried out by the `output` transition. The pallet itself has no different states and is only implicitly modelled together with a work piece.

### 4.3 Performability Evaluation of the Example

After the structure and work plans have been modelled with separate coloured Petri nets as described above, a complete model can be automatically generated by TimeNET [27]. This is done by adding the information contained in the processing sequence models to the structural model. The transitions are enriched with descriptions of their different firing possibilities.

The resulting complete model can then be checked with an interactive simulation (token game) or an automated animation. Structural properties like invariants can be computed. The analysis and simulation algorithms of TimeNET are then started for an evaluation of the system performance, taking into account failures and repairs of the modelled system.

For the application example, the goal is to maximise the throughput of the manufacturing system by adjusting the number of available pallets. This number has usually an important influence on the system performance. The cost of additional pallets is not taken into account. However, it would be possible to include measures of profits and costs in the model, to assess the overall gain. The production mix is set to 50% A and 50% B. A performance measure is defined in the model which gives the throughput of all finished parts per hour.

A deterministic firing delay is associated to most of the transitions in the application example. The reason for this is that simple operations of an automated system like in this case do not vary significantly. Therefore the restriction of at most one enabled transition with non-exponentially distributed firing delay in each marking is violated. The simulation component is thus used for the performance evaluation. All evaluations have been carried out on a cluster of ten UltraSparc workstations with a confidence interval of 99% and a maximum

relative error of 5%. Each simulation run typically took 20 seconds real time and 125 seconds overall CPU time to complete.

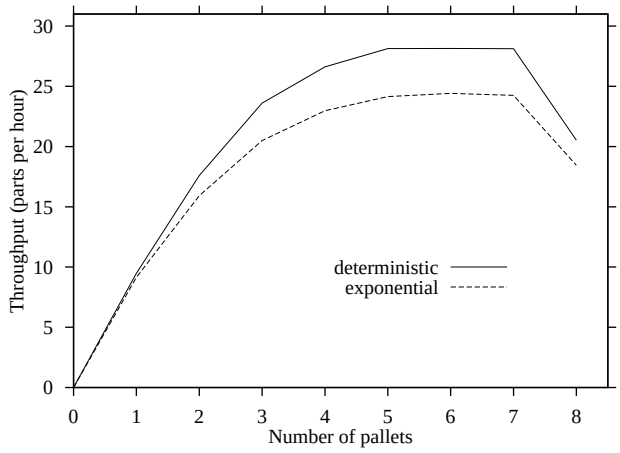


**Fig. 5.** Throughput of the system versus number of pallets

TimeNET computes all results automatically with the experiment feature described in section 3. Figure 5 shows the resulting graph (upper plot). In the range of zero to three pallets, the throughput increases almost linearly. The number of pallets is the most significant bottleneck of the system in these cases. An optimal behaviour is achieved with five, six or seven pallets. The performance drops if their number is increased to eight, because there are only nine possible locations of pallets in the modelled system. Blocking becomes a problem in this case.

To analyse the influence of the deterministic transition firing times a second model was evaluated. All deterministic delays were substituted by exponential ones with the same mean firing delay. The lower plot in Figure 5 depicts the results. Just like in the deterministic case, a number of pallets between five and seven is optimal. Otherwise the absolute values are quite different, showing the importance of deterministic firing delays for a precise performance prediction. Much effort is still needed in the development of efficient analysis techniques for models with non-exponentially distributed firing delays.

## 5 Conclusions

This paper described the software package TimeNET 3.0, which supports the modelling and performability evaluation of discrete event systems. Different

classes of Petri net models are available, namely Markov regenerative stochastic Petri nets and their coloured extension. The latter is especially used for the separate modelling of manufacturing systems and processing sequences. Efficient numerical analysis methods are implemented for the calculation of performance measures either in steady-state or up to a transient point in time. Basic structural properties are detected. An efficient simulation module is available, which takes advantage of parallel replications, rare event simulation, and control variates. The easily adaptable graphical interface integrates the model specification with the access to analysis modules for all net classes. Usage of TimeNET 3.0 for the modelling and performance evaluation of a manufacturing system is explained in the paper with an application example.

The tool is available free of charge for non-profit use. It has been successfully used in many projects and was distributed to more than 150 universities and other organisations worldwide. TimeNET runs under Solaris and Linux.

In the future we plan several extensions of TimeNET: implementation of transient analysis and simulation for coloured models, a new structured modelling method based on Petri nets (Stochastic Petri Net Language), implementation of numerical analysis methods for fluid stochastic Petri nets, and methods for efficient performance analysis of large systems.

The authors would like to thank the former colleagues and numerous students who have contributed to the development and implementation of the software package TimeNET.

## References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Series in parallel computing. John Wiley and Sons, 1995.

[2] F. Bause, P. Buchholz, and P. Kemper. QPN-Tool for the Specification and Analysis of Hierarchically Combined Queueing Petri Nets. In H. Beilner and F. Bause, editors, *Quantitative Evaluation of Computing and Communication Systems*, volume 977, pages 224–238. Springer Verlag, 1995.

[3] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Perf. Eval.*, 24:47–68, 1995.

[4] H. Choi, V. G. Kulkarni, and K. S. Trivedi. Markov Regenerative Stochastic Petri Nets. *Performance Evaluation*, 20:337–357, 1994.

[5] G. Ciardo, J. Muppala, and K. S. Trivedi. SPNP: Stochastic Petri Net Package. In *Proc. 3rd Int. Workshop on Petri Nets and Performance Models*, pages 142–151, Kyoto, Japan, 1989.

[6] G. Ciardo, R. German, and Ch. Lindemann. A Characterization of the Stochastic Process Underlying a Stochastic Petri Net. *IEEE Transactions on Software Engineering*, 20:506–515, 1994.

[7] G. Ciardo and R. Zijal. Well-Defined Stochastic Petri Nets. In *Proc. 4th Int. Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'96)*, pages 278–284, San Jose, CA, USA, 1996.

[8] R. German and J. Mitzlaff. Transient Analysis of Deterministic and Stochastic Petri Nets with TimeNET. In *Proc. Joint Conference 8th Int. Conf. on Modelling Techniques and Tools for Performance Evaluation*, volume 977 of *Lecture Notes in Computer Science*, pages 209–223. Springer Verlag, 1995.

[9] R. German. *Analysis of Stochastic Petri Nets with Non-Exponentially Distributed Firing Times.* Dissertation, Technische Universität Berlin, 1994.

[10] R. German, Ch. Kelling, A. Zimmermann, and G. Hommel. TimeNET – a Toolkit for Evaluating Non-Markovian Stochastic Petri Nets. *Perf. Eval.*, 24:69–87, 1995.

[11] R. German and Ch. Lindemann. Analysis of Stochastic Petri Nets by the Method of Supplementary Variables. *Perf. Eval.*, 20:317–335, 1994.

[12] A. Heindl and R. German. A fourth Order Algorithm with Automatic Stepsize Control for the Transient Analysis of DSPNs. *IEEE Trans. Softw. Engin.*, 25:194–206, 1999.

[13] Ch. Kelling. Rare Event Simulation with RESTART in a Petri Net Modeling Environment. In *Proc. of the European Simulation Symposium*, pages 370–374, Erlangen, 1995.

[14] Ch. Kelling. *Simulationsverfahren für zeiterweiterte Petri-Netze.* Dissertation, Technische Universität Berlin, 1995. Advances in Simulation, SCS International.

[15] Ch. Lindemann. *Performance Analysis of Complex Systems by Deterministic and Stochastic Petri Net Models.* Dissertation, Technische Universität Berlin, 1992.

[16] Ch. Lindemann. *Stochastic Modeling using DSPNexpress.* Oldenbourg, 1994.

[17] Ch. Lindemann. DSPNexpress: A Software Package for the Efficient Solution of Deterministic and Stochastic Petri Nets. *Performance Evaluation*, 22:3–21, 1995.

[18] J.F. Meyer. On Evaluating the Performability of Degradable Computing Systems. *IEEE Trans. Comput.*, 22:720–731, 1980.

[19] J.K. Muppala, G. Ciardo, and K.S. Trivedi. Stochastic Reward Nets for Reliability Prediction. *Comm. Reliab., Maintenab. and Serviceab.*, 1(2):9–20, 1994.

[20] W. H. Sanders, W. S. Obal II, M. A. Qureshi, and F. K. Widjanarko. The Ultrasan Modeling Environment. *Performance Evaluation*, 24:89–115, 1995.

[21] M. Silva and E. Teruel. Petri Nets for the Design and Operation of Manufacturing Systems. *European Journal of Control*, 3(3):182–199, 1997.

[22] R. Zijal. *Analysis of Discrete Time Deterministic and Stochastic Petri Nets.* Dissertation, Technische Universität Berlin, October 1997.

[23] A. Zimmermann, S. Bode, and G. Hommel. Performance and Dependability Evaluation of Manufacturing Systems Using Petri Nets. In *1st Workshop on Manufacturing Systems and Petri Nets, 17th Int. Conf. on Application and Theory of Petri Nets*, pages 235–250, Osaka, Japan, 1996.

[24] A. Zimmermann, K. Dalkowski, and G. Hommel. A Case Study in Modeling and Performance Evaluation of Manufacturing Systems Using Colored Petri Nets. In *Proc. of the 8th Europ. Simul. Symp.*, pages 282–286, Genoa, Italy, 1996.

[25] A. Zimmermann and J. Freiheit. TimeNETms — an Integrated Modeling and Performance Evaluation Tool for Manufacturing Systems. In *IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 535–540, San Diego, USA, 1998.

[26] A. Zimmermann, A. Kühnel, and G. Hommel. A Modelling and Analysis Method for Manufacturing Systems Based on Petri Nets. In *Comput. Engineer. in Systems Appl. (CESA '98)*, pages 276–281, Nabeul-Hammamet, Tunisia, 1998.

[27] A. Zimmermann. *Modellierung und Bewertung von Fertigungssystemen mit Petri-Netzen.* Dissertation, Technische Universität Berlin, September 1997.