TimeNET$_{MS}$ – An Integrated Modeling and Performance Evaluation Tool for Manufacturing Systems

Armin Zimmermann and Jörn Freiheit
Technische Universität Berlin, Institut für Technische Informatik
Franklinstr. 28/29, Sekr. FR 2-2, 10587 Berlin, Germany

ABSTRACT

There is a need for modeling and performance evaluation techniques and tools for a fast and reliable design of modern manufacturing systems. This paper introduces TimeNET$_{MS}$, a software tool which integrates manufacturing system modeling and performance analysis with an easy-to-use graphical user interface. It is based on a concept of independent models for the manufacturing system structure and work plans, both using a dedicated class of colored stochastic Petri nets. TimeNET$_{MS}$ is implemented as an extension of the more general Petri net modeling and analysis tool TimeNET. An industrial application example shows the use of the tool.

1 INTRODUCTION

Complex manufacturing systems require the use of computers and software tools throughout their life cycle. This fact is now well accepted for the online control of manufacturing systems. But there are still industrial projects where the design is done without proper computational methods and tools. The results are design errors that could have been avoided using a modeling and analysis tool.

On the other hand, there are several modeling and analysis techniques available that have been developed mainly in universities. Among them, Petri nets with stochastic timing play a major role in the application area of manufacturing systems [8].

Integrated tools that are easy to use are necessary to make those theoretical results applicable in industry. That includes an up-to-date graphical user interface as well as application domain specific modeling features. The analysis and simulation techniques being used have to be robust and fast.

This paper describes the new modeling and performance analysis tool TimeNET$_{MS}$ (TimeNET for manufacturing systems), which implements the modeling and analysis techniques described in [10]. The tool is a new extension of the more general modeling and performance evaluation tool TimeNET (**Time**d **N**et **E**valuation **T**ool, [4]). TimeNET uses non-Markovian stochastic Petri nets without colors for the modeling and analysis of discrete systems.

The class of colored stochastic Petri nets used by TimeNET$_{MS}$ has been introduced especially for the modeling of manufacturing systems. Two color types are predefined: *Object tokens* model workpieces 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 uncolored Petri nets. Places can contain only tokens of one type. The model of the manufacturing cells structure reflects the layout, which makes it easier to understand. Textual descriptions needed in colored Petri nets for the definition of variables and color types can be omitted, and the specification of the types of places and arcs are implicitly given. TimeNET$_{MS}$ supplies a library of template models for typical machines and their failure-and-repair behavior. The models are hierarchically structured, which is necessary to handle complex systems.

Structure and work plans are modeled 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 route 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 synchronization of the production steps.

Finally, measures of interest are obtained from the complete model. The correct behavior of the modeled system can be checked with an interactive simulation (token game). Quantitative properties are derived by numerical analysis or simulation, showing the performance and dependability of the considered system. The design process becomes more reliable with the opportunity to calculate system properties. An optimization is possible by evaluating and comparing measures of interest for different model variations.

Several software tools are related to TimeNET$_{MS}$. DesignCPN supports colored Petri nets [5] without being specialized for an application area like manufacturing systems. It is thus more powerful but not as easy to use for the modeling of manufacturing systems. As DesignCPN is not intended for performance evaluation of stochastically timed models, it does not offer the necessary analysis and simulation techniques.

XPN-FMS is a tool for the modeling, structural analysis, and interactive simulation of manufacturing systems [1]. However, it is limited to decision-free Petri nets without distinguishable tokens. This makes its application to real-life manufacturing systems practically impossible.

The tool GRAMAN [9] offers separate modeling of production routes and manufacturing system structure. While for the work plans colored Petri nets are used, the structure of the system is specified with predefined building blocks. An internal model is generated from these two descriptions. The connection of submodels that are created from the building blocks during this translation is done by fusing transitions that represent synchronised activities. Unfortunately, this means that those connections have to be known at the time when the submodel is specified. This somehow contradicts the modularity of the specification. The modeler has to learn two different description languages before using the tool. Additionally, it is not possible to specify machine properties that depend on a processing task.

The paper is organised as follows. Section 2 introduces the concepts of the graphical user interface of TimeNET$_{MS}$. In section 3 the computation of performance measures by the tool is described. The subsequent section illustrates the application using an example. In the final section concluding remarks are given.

2 A GENERIC GRAPHICAL USER INTERFACE

A graphical interface is necessary for user-friendly specification and analysis. The applied modeling and analysis method makes use of several different model types. There are special model classes for the manufacturing system structure, the work plans, the complete model, library templates, and so on.

We thus had to develop a user interface capable of a number of model types. The existing interface of TimeNET 2.0 was tailor-made for uncolored Petri nets such as extended deterministic and stochastic Petri nets. To include the new net classes and to be able to adapt to future extensions easier, we decided to implement a generic graphical user interface. It is called Agnes (**a** **g**eneric **n**et **e**diting **s**ystem) [7] and is already used for other tools besides TimeNET$_{MS}$. This interface can be used for graphs with different node and arc types. Nodes can be hierarchically refined by underlying submodels. Agnes is implemented with C++ and Motif.

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. Each net class 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. Attributes have types and identifying names, which can be basic (e.g. string, integer, real and flag), lists and structures of other types. The shape of each node and arc is defined using a set of primitives (e.g. polyline, ellipse, and text). Line thickness, color, line style, font and other shape attributes can be specified as well. Using shapes that depend on attribute values of objects, it is for instance possible to show tokens as dots inside places.

When the user interface is started, it generates the necessary windows and icons from the textual net class definitions. Without any programming the user then has a graphical interface for his model class, with all editing functions like save, load, undo, print, cut, paste, and so on.

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. Every time Agnes opens or creates a new window, it calls each module. The module can then tell the interface, whether it is applicable for the current net class or not. Additionally,

it can insert menu entries or add more complicated functions. If the menu item is clicked by the user, the corresponding module is called and can start a simulation process for instance. Modules are therefore free to either implement functions for one, several or all specific net classes. An example of an implemented module applicable without restrictions is an export filter to *xfig*. Modules can access the model through an interface to Agnes.

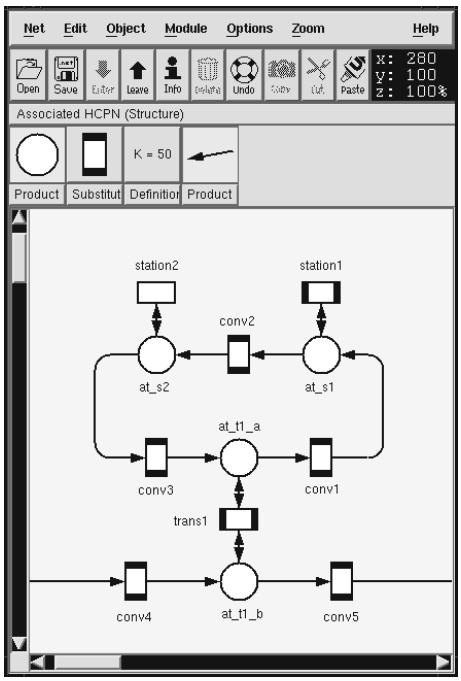

Figure 1: Screenshot of the graphical user interface

Figure 1 shows a sample screenshot of TimeNET$_{\mathrm{MS}}$. The upper row contains some menus with usual actions for file access, editing, and display options. The most frequently used functions are also available by clicking on one of the icons below. The third row displays the current net class. Below that, the set of model elements belonging to the net class are displayed on buttons. Each one is given as a picture icon and an identifier. Objects are grouped on buttons.

The main drawing area shows a part of the current model. It can be edited with the mouse like using a drawing tool. Double-clicking on an object brings up a window with all attribute identifiers and current values of that object, like the firing delay for a transition.

3 PERFORMANCE EVALUATION WITH TimeNET$_{\mathrm{MS}}$

After specifying a manufacturing systems model using the described 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. Currently we are working on their development and later implementation for the dedicated class of colored Petri nets. A second debugging tool is the token game, showing the states and actions of the system. With TimeNET$_{\mathrm{MS}}$ the modeler is able to either interactively check the model behavior or to watch an automatic animation of it. However, the main focus is on efficient quatitative evaluation of performance and dependability measures. Throughput, utilisation, and availability are typical examples for manufacturing system measures.

Firing times are associated with transitions for the performance evaluation. We adopt the set of distributions as defined for extended deterministic and stochastic Petri nets (eDSPN) here [3]. They include immediate, exponential, deterministic, and more general transitions.

There are two possible methods for the computation of performance measures, direct numerical analysis and discrete event simulation. Depending on the model and the measures to be evaluated, both have their advantages and disadvantages. Direct numerical analysis gives exact results and is often faster than a simulation. However, if more than one transition with non-exponentially distributed firing time is enabled in one marking, or if the state space is too large to be computed, simulation is the only choice. As long as at most one of those transition might fire in each marking, the underlying stochastic process belongs to the class of continuous-time semi-regenerative Markov processes [2]. For this class numerical analysis techniques are known [3], which have been adapted to the net class used here and implemented in TimeNET$_{\mathrm{MS}}$.

Figure 2 shows the interaction of the user interface and other processes during a direct numerical analysis. When the user clicks on the corresponding menu item, an Agnes module (see section 2) writes the model description to a file and invokes the subsequent processes. The reachability graph is computed in the first step. Afterwards, the one-step probability matrix P and the matrix of mean sojourn times C

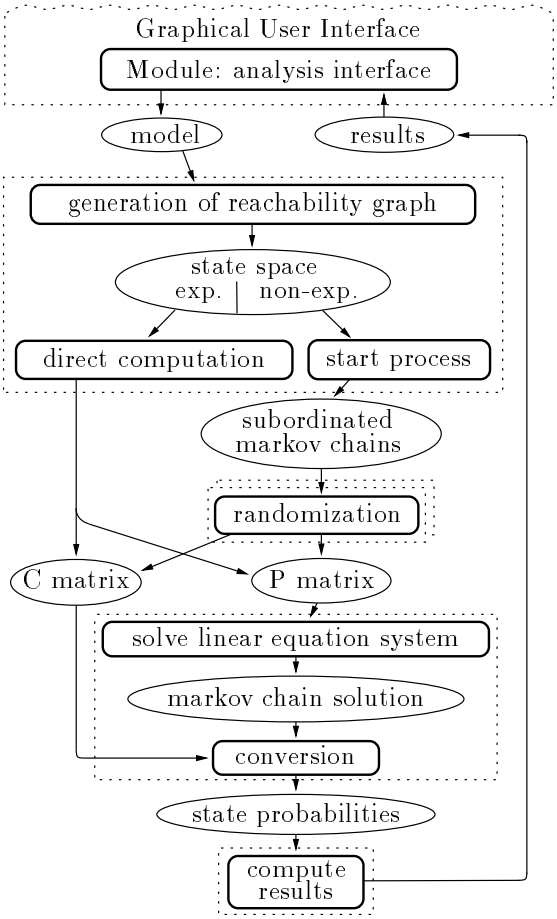

Figure 2: Direct numerical analysis

have to be computed. For markings without enabled non-exponential transitions, this can be done directly. Other markings require a transient analysis of a subordinated stochastic processes. The corresponding matrix entries are computed in parallel using randomization (Jensens method). The system of linear balance equations given by P (embedded Markov chain) is then solved, and the C matrix is used to convert the result into the time domain. Performance measures can be computed from the normalized resulting vector of steady-state marking probabilities.

For an efficient parallel discrete-event simulation a master/slave concept is used [6]. Figure 3 shows the process interactions. The master process starts several replications of slave processes on different workstations. Each slave has a copy of the whole model and simulates its behavior. Packets of measure samples are sent to the master, who uses spectral variance analysis to check whether the specified accuracy is reached. The confidence level and maximum relative error of the measures can be set. The initial transient phase of a simulation run is automatically detected and ignored.

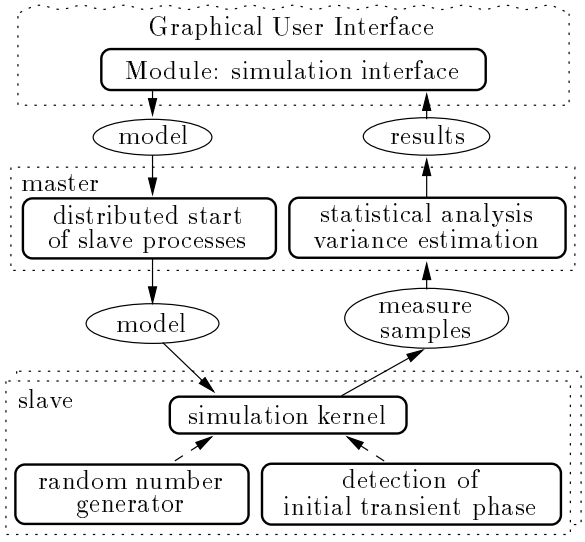

Figure 3: Distributed simulation

4 AN APPLICATION EXAMPLE

An industrial application example is presented in this section to explain the use of TimeNET$_{MS}$. The selected example is an assembly system producing buckle pretensioners for car safety belts. This example is used here only to explain the software tool without going into the model details. Further information on the manufacturing system can be found in [11].

Twenty different parts have to be assembled. Most of the 17 stations in the system operate fully automatic, there are only three manual workplaces. The different stations are organized in three circular assembly lines, that the parts have to pass one after another. Each circle consists of several conveyor belts and conveyor switches, that connect the different stations and act as buffers.

The drawing area of the tool shown in Figure 1 contains a part of the structural model. It describes the first assembly circle of the manufacturing system. Transitions model assembly stations and conveyor belts. Places are used for buffers and processing places at assembly stations.

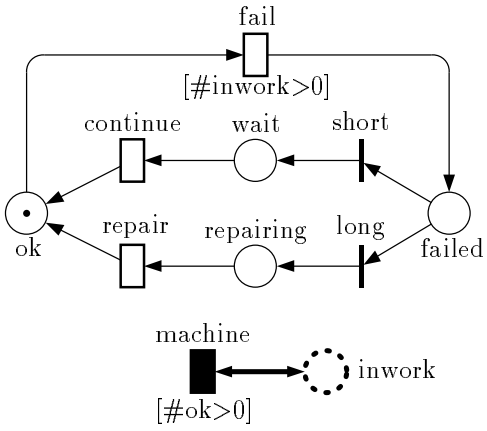

Figure 4: Submodel with failure behavior

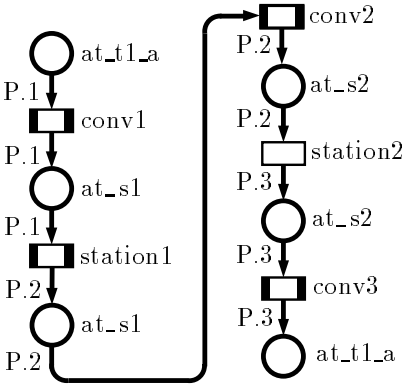

Figure 5: Production route model

Each substitution transition (depicted as ▯) is refined by a subpage that describes the behavior of the resource with more detail. Submodels from a library of standardized building blocks (*templates*) can be parameterized and instantiated while refining the model. Each template has a set of parameters such as processing times or buffer capacities. By associating values to the parameters, each of them represents a class of structurally similar resources. They can be used for the refinement of substitution transitions. In our example, the failure and repair behavior of all assembly stations could be described with a submodel as it is shown in Figure 4. Different firing times had to be set for the transitions of each station model.

A production route model for each workpiece has to be defined then. Figure 5 shows the top level of this model for parts in the first assembly circle. This model describes the sequence of processing steps and transports for one part. Only elements of the corresponding structural model (identified by their names) can be used here. The arcs are labelled with the names of parts, showing the changes in their processing state. The name of a part and its processing state are separated by a dot.

Alternative routes of workpieces can be modeled using different paths, with conditions and probabilities assigned to them if needed. However, this was not necessary for the work plan presented here. Conditions decide which path might be chosen, thus allowing the implementation of a scheduling strategy. Usually, a work plan model consists of a linear sequence of places and transitions. An exception are — in addition to alternative routes — assembly and disassembly operations, where more than one input or output arcs are connected to a transition.

After the structure and the work plans have been modeled with the software tool, different analysis methods can be applied. The models are automatically merged to create a complete model. During this process, the information contained in the production route models are added to the structural model. This is done by an Agnes module.

The resulting complete model can then be checked with an interactive simulation or an automated visual animation. Afterwards, performance measures of the modeled system can be computed using direct numerical analysis or parallel discrete event simulation (as described in section 3).

Often it is desirable to evaluate the system performance for different parameter values in some given range. The *experiment* module of TimeNET$_{\mathrm{MS}}$ allows the specification and automatic computation of such a problem. The parameter can be varied linearly or logarithmically. *Gnuplot* is started after the computation to display the results graphically.

Each part inside an assembly circle is transported on a carrier. An important question is thus the influence of the number of carriers on the performance of the assembly system. Figure 6 shows the throughput measured in parts per hour of the assembly line, if the number of part carriers is varied from 0 to 95. The throughput increases if the number of carriers is raised in the range up to 40 carriers. The upper and lower curves in Figure 6 show the impact of machine failures on the total performance. The optimal number of available carriers can be obtained depending on these results.

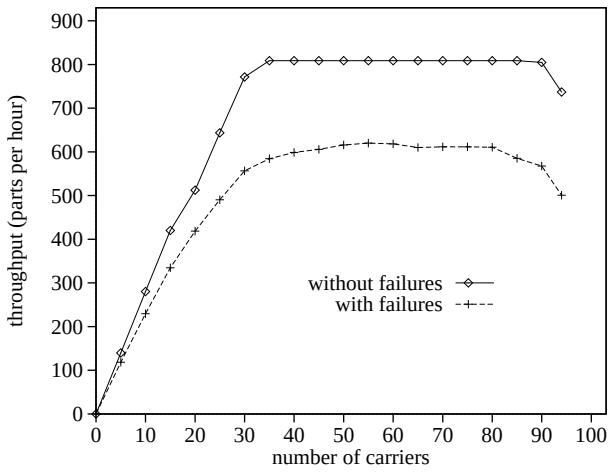

Figure 6: Throughput versus number of carriers

5 CONCLUSION

This paper describes TimeNET$_{MS}$, a new software tool for the modeling and performance analysis of manufacturing systems. A special class of colored Petri nets is used to specify independent models for the system structure and the work plans. State-of-the-art numerical analysis and simulation techniques are implemented for the performance analysis. The graphical user interface is generically implemented and can thus be used for many types of graph-like models. Usage of the tool has been demonstrated with an industrial application example.

TimeNET$_{MS}$ and Agnes are available free of charge for non-commercial use as part of the tool TimeNET 3.0. They require a SparcStation under Solaris; a Linux port is under development. Further information can be found at `http://pdv.cs.tu-berlin.de/~timenet` and `http://pdv.cs.tu-berlin.de/~agnes`.

The authors would like to acknowledge the work of the master students Frank Bayer, Mario Beck, Kolja Koischwitz, Andreas Kühnel, and Ken Schwiersch during the implementation of TimeNET$_{MS}$.

REFERENCES

[1] D. Y. Chao and D. T. Wang, "XPN-FMS: A CAD Tool for FMS Modeling, Analysis, Animation, and Simulation Using Petri Nets and X Window", *Int. J. of Flexible Manufacturing*, No. 7, 1995, pp. 339–360.

[2] G. Ciardo, R. German, and C. Lindemann, "A Characterization of the Stochastic Process Underlying a Stochastic Petri Net", *IEEE Transactions on Software Engineering*, No. 20, 1994.

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

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

[5] K. Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, EATCS Monographs on Theoretical Computer Science, Springer Verlag 1992).

[6] C. Kelling, *Simulationsverfahren für zeiterweiterte Petri-Netze*, Ph. D. Dissertation, Technische Universität Berlin, 1995.

[7] K. Koischwitz, *Entwurf und Implementierung einer parametrisierbaren Benutzungsoberfläche für hierarchische Netzmodelle*, Master's thesis, Technische Universität Berlin, October 1996.

[8] M. Silva and E. Teruel, "Petri Nets for the Design and Operation of Manufacturing Systems", in: *Proc. 5th Int. Conf. on Computer Integrated Manufacturing and Automation Technology (CIMAT '96)*, Lille 1996.

[9] J. L. Villaroel, J. Martínez, and M. Silva, "GRAMAN: A Graphic System for Manufacturing System Design", in: S. Tzafestas, ed., *IMACS Symposium on System Modelling and Simulation*, 1989, pp. 311–316.

[10] 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*, Osaka 1996, pp. 235–250.

[11] 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 European Simulation Symposium*, Genoa 1996, pp. 282–286.