Tool Support for Model-Based Online Control of Manufacturing Systems

Armin Zimmermann and Jörn Freiheit
Institut für Technische Informatik
Technische Universität Berlin, Sekr. FR 2-2
Franklinstr. 28/29, 10587 Berlin, Germany
phone: +49 (30) 314 73 112 fax: +49 (30) 314 21 116
email: {azi, freiheit}@cs.tu-berlin.de

ABSTRACT

Design and operation of automated manufacturing systems is a complex task. It should be supported by the use of a computer model for planning, functional tests, performance evaluation and optimisation. It is advantageous to use the resulting model for the control as well, enabling an integration of the different design tasks using only one common model. This paper describes the control extension of a software tool for the modelling and analysis of manufacturing systems. Colored stochastic Petri nets are used for the system description.

Keywords: Modelling, Manufacturing Systems, Control, Petri Nets, Software Tool

1. INTRODUCTION

Modern complex manufacturing systems require methodological support throughout their design and operation. Petri nets [13, 15, 12] have been successfully applied in this field, due to their ability to describe systems exhibiting concurrency, conflicts, and synchronisation [3, 4, 14, 17, 16]. This necessitates software tools that implement the methodologies developed for Petri nets in the application area of manufacturing systems. The paper describes a recent enhancement of the modelling and performance evaluation tool TimeNET [20], that allows the online control of manufacturing systems. The new part is integrated into the manufacturing systems modelling and evaluation environment of the tool [19, 21]. The aim is to offer support throughout the different life cycle phases of a manufacturing system. The same model (undergoing changes and becoming more and more detailed) is used from the first design steps until the online control. A special class of colored Petri nets is used for the modelling, facilitating seperate models of system structure and work plans [18].

2. RELATED WORK

Petri nets have been often considered for control of manufacturing systems [7]. Some approaches to interprete a Petri net model for control use different methods to exchange information between model and process. Sometimes the token contents of places (the marking) is used to generated the model output. This is e.g. the case in GRAFCET [1, 2] and related methods. The enabling of transitions that depend on external information can also be done with *control places* [8]; the models are then called controlled Petri nets. Another possibility is the association of control procedures with transitions [11]. Colored Petri nets are used for the control of a manufacturing system e.g. in [9, 10]. In [5], transitions modeling processing steps are hierarchically refined and input/output signals are associated with the subtransitions. Generally, control design based on a Petri net model is advantageous with respect to a state machine description, because they are able to capture parallelism in a much clearer way.

3. APPLICATION EXAMPLE

This section briefly describes the application example, a simple manufacturing cell. Figure 1 shows its layout.

In the considered production cell, new work pieces are initially stored in the high bay racking on pallets. The rack conveyor can fetch one of them and deliver it to the upper pallet exchange place. A horizontal crane then takes it to the first conveyor belt. Three conveyor belts moves work pieces from one processing station to another. There are two drilling stations, the second having three different interchangeable drilling tools. The last station is a milling machine. Work pieces stay on the conveyor during processing. After leaving the machines, they

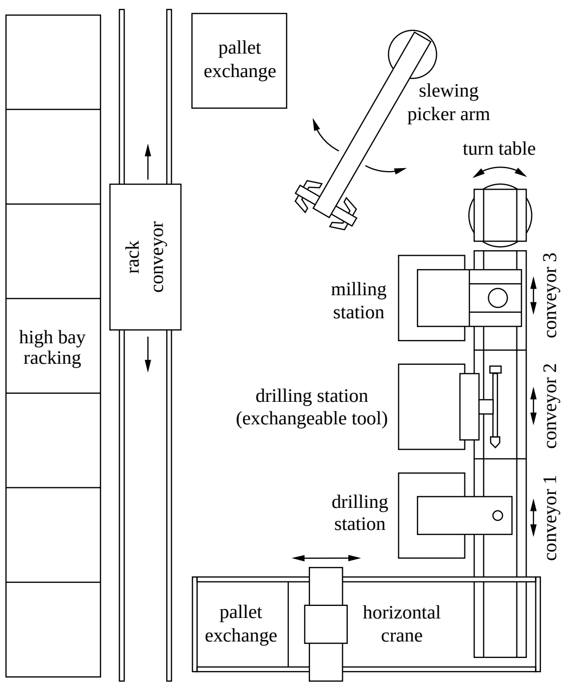

Figure 1: Overview of the modeled system

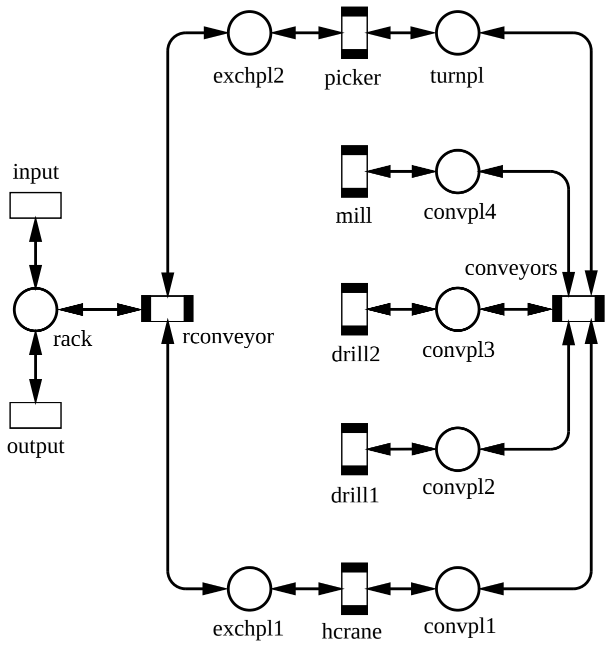

Figure 2: Highest level of the hierarchical model

arrive at a turn table. This table puts them into position for the slewing picker arm, who takes the work piece to the right 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 work pieces takes place via the rack storage. Throughout the paper it is assumed that work pieces have to be machined by the two drilling machines and the milling machine in this order. The work pieces are moving counterclockwise through the system.

The described application example is modeled with a special class of dedicated colored Petri nets [21]. A strict separation between the model of the manufacturing system's structure and the sequence of the processing steps for each product is observed.

Figure 2 shows the top level of the structural model. Its composition follows the layout of the modeled system, which makes it easier to understand. Places model buffers and other possible locations of work pieces. 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 work pieces takes place through the rack storage and is modeled with transitions `input` and `output`.

In principle, 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 modeled by a change in the color of the token corresponding to the work piece. Transitions modeling machines specify processing steps which only change the token color. This is emulated by removing the former token from the place and instantly adding a token with the new color by firing the transition. Therefore many transitions and places are connected by arcs in both directions (loops), 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 work pieces from the conveyor to the exchange place as well.

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 behavior of a machine with more detail during a top-down design. Submodels from a library of standardized building blocks (templates) can be parameterized 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, respectively. For the transport of a work piece 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 facility and are modeled by transition `conveyors`. Thus, their synchronization is hidden at a lower level and can be specified together. The whole model consists of 11 submodels.

4. CONTROL INTERPRETATION

After the planned manufacturing system has been designed and e.g. its performance evaluated, the final step is to bring the specified behaviour into reality. This can be done by interpreting the model, directly using it for the online control of a production process. Thus only one model is used throughout the whole design process. It should however be noted that the aim of this paper is not synthesis, PLC program generation, nor verification of control strategies. The purpose is rather to show that the used class of Petri nets is applicable for the different life cycle phases without major changes in the model. It is easily possible to integrate control rules in the model (e.g. by adding marking dependent firing guards to transitions). Afterwards, the influence on the system behaviour (lifeness, performance) can be analysed. Finally the behaviour specified in the model is directly executed.

To allow the control of a real-world process using the Petri net model, possibilities for its interaction with the outside world have to be added to the otherwise autonomous model. From the point of view of the model, input and output signals are necessary. They should be added in a simple way, following the meaning of the performance model elements in a natural way. This is possible without problems, because the used kind of colored Petri net models strongly observe the modeled system's structure.

Only "active resources" like machines, transport facilities etc. are controllable. Their activities are modelled by transitions, who can either move tokens (transport) or change token color (processing). A transition becomes enabled, when its guard function evaluates to true, the necessary input tokens are available, and enough space for added tokens is free in the subsequent places. This is the point of time at which the firing time of the transition is running. The actual firing with the corresponding marking change takes places when the firing delay has elapsed. It then appears natural to assign single controllable activities to transitions. When the transition becomes enabled, a start (output) signal is sent from the model to the process (e.g. a motor is switched on). After termination of the activity (e.g. a sensor detects the stop position), an (input) signal from the process is sent to the model, which then initiates the instantaneous transition firing. This model interpretation only changes the model behaviour by adopting the unknown delays of external activities.

Transitions with associated input signals are called *external*, all others *internal*. Associating output signals to a transition does not change its firing semantic in the model. Opposed to that, external transitions fire if and only if they are enabled and receive their input signal. The firing delay being specified in the model for external transition is ignored during the online control. The modeler should be careful with cases in which external transitions can be disabled by the firing of other transitions, because their input signal could then be lost. Conflicts of this type can be automatically detected from the model structure. However, they are not generally forbidden, because there are cases in which this behaviour is useful.

It is possible to assign any number of input and output signals to a transition. All output signals are being sent when the transition becomes enabled, while the arrival of any one of the input signals triggers the firing of the transition. Transitions with output signals but without input signals (or vice versa) are allowed as special cases. An example of an activity which can be finished at any time without having been started before is e.g. the failure of a machine. A sensor which detects this failure can trigger the firing of an associated transition in the model. The firing time of internal transitions keeps its semantics from the autonomous model. During the online control of a production process, the usually unit free numbers have to be interpreted e.g. as seconds. They can be used to control the timing of activities without stop positions and to detect deadline violations.

Motivated by the use of seperate models for structure and work plans of a production system, the task of designing the control system can be divided in two steps. The specification of the work plans ensures that the manufacturing system produces (in the model) the desired products. However, the afterwards automatically generated main model does not necessarily have to be free of deadlocks nor optimal with respect to some performance measures like the throughput. The model can be analysed and the problems detected, leading to control strategies

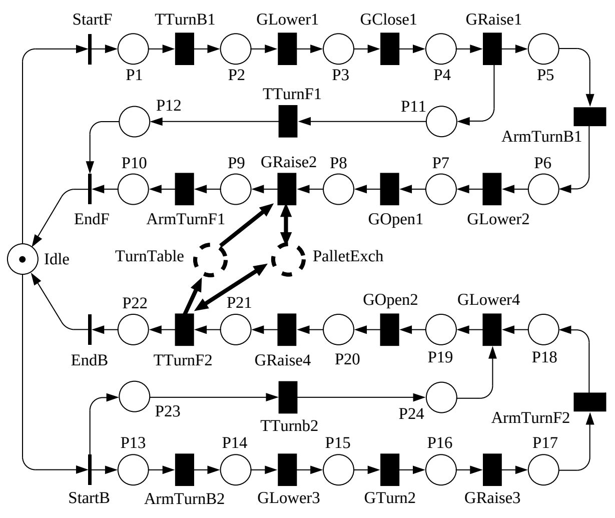

Figure 3: Refined model of the slewing picker arm

that improve the system behaviour. In any state of the production system, the controller can only forbid activities that would otherwise be possible. This corresponds to disabling transitions inside the model. Using marking dependent guard functions, this can be done easily with the used class of Petri nets. The guard function of a transition (or of one firing possibility) is a boolean function of the net marking, which has to be true to allow the enabling. The second step of the control design thus corrects and optimises the production process. Finally, the model based online control ensures the execution of the specified behaviour.

5. CONTROL OF THE EXAMPLE

The Petri net model shown in Figure 3 is a hierarchical refinement of the substitution transition `picker`, that is part of the main structural model (shown in Figure 2). It specifies the inner behavior of the slewing picker arm as well as the correlated control of the turn table. The different steps for the two possible transport actions are described by sequences of elementary transitions and places. Each transition corresponds to one controllable activity. The *system states* of the slewing picker arm are modeled by *elementary* places and arcs (drawn thin in Figure 3). The possible *locations of work pieces* are modeled by the *object* places `TurnTable` and `PalletExch` (drawn thick). Because these places are also visible on the upper model level, they are depicted with dashed circles.

The slewing picker arm can execute two useful actions: take a work piece from the turn table to the upper pallet exchange and the reverse. The current state of the picker arm (and of the turn table) corresponds to the location of the elementary token in the model. Figure 3 shows the state after initialization, where the token is in place `Idle`. Either one of the two immediate transitions `StartF` and `StartB` can fire if the resources are idle, thus starting one of the two transport actions. The decision is made by firing guards (marking dependent boolean expressions) of the transition table entries. It ensures that if the work plan models are correctly specified, the picker arm is only activated for useful transport activities.

Transitions with names beginning with `G` describe actions of the picker arm gripper (lower, close, raise, open). The `ArmTurn` transitions model the turning of the picker arm, and the turn table is described with transitions named `TTurn`.

After the firing of one of the two starting transitions, the corresponding transport action begins. The different transitions become enabled and fire in succession. They describe the individual steps of each transport.

Transition `TTurnF1` is an example for steps that can be executed in parallel to others. Transitions `GRaise2` and `TTurnF2` move the tokens (or change their colors) in places `TurnTable` and `PalletExch`. This models the step after which the work pieces are available for subsequent actions, even if the picker arm still has to be moved back into its default position.

6. SOFTWARE TOOL

Modelling and evaluation of complex systems is only feasible with the support of appropriate software tools. Since the modelling framework of stochastic Petri nets has been proposed, many algorithms and their implementations as software tools have been developed. A powerful and easy to use graphical interface is important in addition. The techniques described in this paper have been implemented in the tool TimeNET (**Time**d **N**et **E**valuation **T**ool, [6, 20]).

For the current version 3.0 of TimeNET a new generic graphical user interface has been developed. Figure 4 shows a sample screen shot of the interface during a modelling session with coloured Petri nets. The interface can be used for graph-like models with different types of nodes and arcs. Nodes can be hierarchically refined by corresponding submodels. It is implemented in C++ and uses the Motif toolkit.

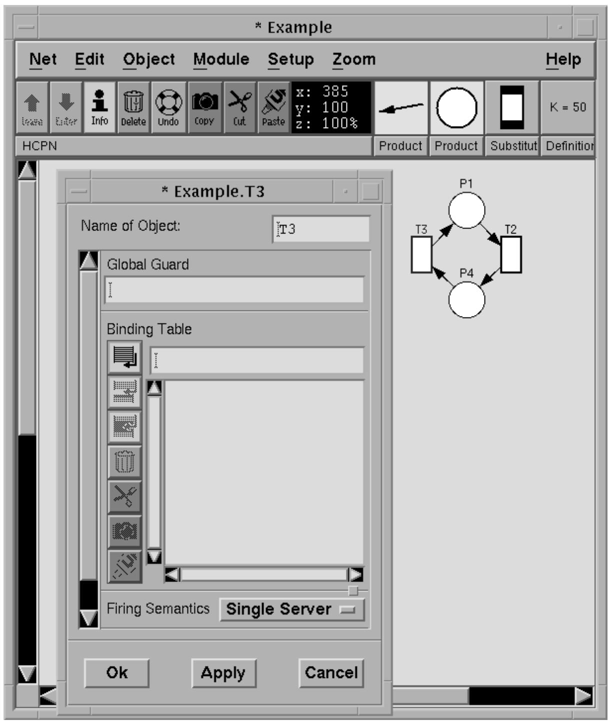
* Example
Net Edit Object Module Setup Zoom Help
Info Delete Undo Copy Cut Paste
x: 385
y: 100
z: 100%
K = 50
HCPN
Product Product Substitut Definitio
* Example.T3
Name of Object: T3
Global Guard
Binding Table
Firing Semantics Single Server
Ok Apply Cancel
P1 T3 T2 P4

Figure 4: Sample tool screenshot

TimeNET is available free of charge for non-commercial use under Solaris and Linux. For further information please refer to the web information at the URL `http://pdv.cs.tu-berlin.de/~timenet`.

7. CONTROL IMPLEMENTATION

With the software tool TimeNET, production processes can be controlled by activating the control during the token game facility. Two variations of it are available: interactive (the user controls the firing of enabled transitions) and animation (simply a simulation). Both show the marking and state changes and can thus be used for online control and visualisation of the production process. The animation can be stopped at any time for manual control. Therefore the control of the modeled production process is seamlessly integrated in the modeling and evaluation tool.

The described functionality is implemented in TimeNET in the following way: During a token game with activated control, an additional process monitors the model state. If a state change enables a transition, its possible output signals are sent to the production process. The sensor states are checked additionally by this process. If a sensor state change is detected, which is associated to an input signal, the corresponding transition(s) are fired if they are enabled in the model and the displayed model state is updated. Internal transitions may fire independently from input signals, only depending on their associated firing time. There is a file describing the correspondance between transitions and signals for models of controllable systems.

8. CONCLUSION

Petri nets are a useful modeling formalism for the description, evaluation, and control of manufacturing systems. In the paper, a recent enhancement of the manufacturing systems part of the tool TimeNET is presented, that allows the online control of manufacturing systems. This enables an integrated support during the different life cycle phases of a manufacturing system. No change to another model or tool is needed, which would cause additional work and many possible errors.

REFERENCES

[1] R. David, GRAFCET – a powerful tool for specification of logic controllers, *IEEE Trans. on Control Systems Technology* **3** (3) (1995) 253–268.

[2] R. David and H. Alla, *Petri Nets and Grafcet (Tools for modelling discrete event systems)* (Prentice Hall, 1992).

[3] A. A. Desrochers, *Modeling and Control of Automated Manufacturing Systems* (IEEE Computer Society Press, Washington, DC, USA, 1990).

[4] F. DiCesare, G. Harhalakis, J. M. Proth, M. Silva, and F. B. Vernadat, *Practice of Petri Nets in Manufacturing* (Chapman and Hall, London, 1993).

[5] K. Feldmann, W. Colombo, and C. Schnur, An Approach for Modeling, Analysis and Real-Time Control of Flexible Manufacturing Systems Using Petri Nets, in: *Proc. European Simulation Symposium* (Erlangen-Nürnberg, 1995) 661–665.

[6] R. German, C. Kelling, A. Zimmermann, and G. Hommel, TimeNET – A Toolkit for Evaluating Non-Markovian Stochastic Petri Nets, *Performance Evaluation* **24** (1995) 69–87.

[7] L. E. Holloway, B. H. Krogh, and A. Giua, A Survey of Petri Net Methods for Controlled Discrete Event Systems, *Discrete Event Dynamic*

Systems: Theory and Applications **7** (1997) 151–190.

[8] L. E. Holloway and B. H. Krogh, Controlled Petri Nets: A Tutorial Overview, in: G. Cohen and J.-P. Quadrat, eds., *11th Int. Conf. on Analysis and Optimization of Systems*, *Lecture Notes in Control and Information Sciences*, Vol. 199, (Sophia-Antipolis, 1994, Springer-Verlag) 158–168.

[9] E. Kasturia, F. DiCesare, and A. A. Desrochers, Real Time Control of Multilevel Manufacturing Systems Using Colored Petri Nets, in: *Proc. Int. Conf. on Robotics and Automation* (1988) 1114–1119.

[10] J. Martínez, P. Muro, and M. Silva, Modeling, Validation and Software Implementation of Production Systems Using High Level Petri Nets, in: *Proc. Int. Conf. on Robotics and Automation* (Raleigh, North Carolina, 1987) 1180–1185.

[11] J. Martínez and M. Silva, A language for the description of concurrent systems modelled by coloured Petri nets: Application to the control of flexible manufacturing systems, in: *Proc. of the 1984 IEEE Workshop on Languages for Automation* (New Orleans, 1984) 72–77.

[12] T. Murata, Petri Nets: Properties, Analysis and Applications, *Proceedings of the IEEE* **77** (4) (1989) 541–580.

[13] J. L. Peterson, *Petri Net theory and the modeling of systems* (Prentice Hall, Englewood Cliffs, New Jersey, 1981).

[14] J.-M. Proth and X. Xie, *Petri nets: A tool for design and management of manufacturing systems* (John Wiley and Sons, 1996).

[15] W. Reisig, *Petri nets* (Springer Verlag Berlin, 1985).

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

[17] N. Viswanadham and Y. Narahari, *Performance Modeling of Automated Manufacturing Systems* (Prentice-Hall Inc., 1992).

[18] 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, Japan, 1996) 235–250.

[19] 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* (San Diego, USA, 1998) 535–540.

[20] A. Zimmermann, J. Freiheit, R. German, and G. Hommel, Petri Net Modelling and Performability Evaluation with TimeNET 3.0, D. Haverkort et.al.: *Lecture Notes in Computer Science*, Vol. 1786, 11th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation (Schaumburg, Illinois, USA, 2000) 188–202.

[21] A. Zimmermann and G. Hommel, Modelling and Evaluation of Manufacturing Systems Using Dedicated Petri Nets, *Int. Journal of Advanced Manufacturing Technology* **15** (1999) 132–137.