RAPID PROTOTYPING OF SEQUENTIAL CONTROLLERS WITH PETRI NETS

Philipp Orth [*] **Alexander Bollig** [*] **Dirk Abel** [*]

[*] *Institute of Automatic Control, RWTH Aachen University, D-52056 Aachen, Germany*

Abstract: This paper focuses on the possibilities of a Petri net based rapid control prototyping (RCP) tool chain. P/T nets extended by interfaces and connected to an extended state model allow the description of typical programmable logic controller tasks. After a general introduction to RCP, the method is applied to the area of sequential control, and code generation for the chosen model is depicted. The paper closes with an example for a tool chain, demonstrated by system simulation and Software-in-the-Loop for a subsystem of a flexible manufacturing system in an industrial fieldbus environment. *Copyright ©2005 IFAC*

Keywords: discrete-event systems, Petri nets, hybrid, modelling, simulation, prototyping, programmable logic controller, sequential control

1. INTRODUCTION

Industrial processes are often controlled by programmable logic controllers (PLCs) with a sequential control algorithm, for which the engineering tools are stunningly lacking several analogous methods compared with continuous control design. On the one hand, advances made in the area of formal discrete event systems analysis are not reflected in the engineering tools which holds for modern synthesis methods as well. On the other hand, only rudiments of the rapid prototyping idea can be found in practice, as most design of discrete control is done without a formal process model. This contribution aims at the development of a rapid prototyping environment for discrete controllers in industrial applications, merging advances in theory with engineering tools.

Conventionally the process description is an informal specification, verbally and graphically expressed and consisting of a mix of control algorithm requirements with depictions of uncontrolled and desired process behaviour (Lunze, 2002). Rapid Control Prototyping (RCP) is based on a formal description of the uncontrolled process for every possible behaviour and a separate model of the control solution. To enable the proposed RCP tool chain, a Petri net based formal model was chosen, which allows for integration of analysis and synthesis methods into the chain.

The paper is organised as follows. In section 2 the model used is described. Section 3 depicts RCP in general, while the next section deals with its application to sequential control. Section 5 demonstrates practical aspects for an example of a flexible manufacturing system.

2. MODEL

This article deals with processes of continuous, discrete or hybrid nature which are controlled by discrete event systems. One of several modelling possibilites for the controller is to use Place/Transition nets (P/T nets) as discrete event subsystems (Abel, 1990; David and Alla, 1994; Antsaklis and Koutsoukos, 1998; Orth *et al.*, 2002). The process is connected to the con-

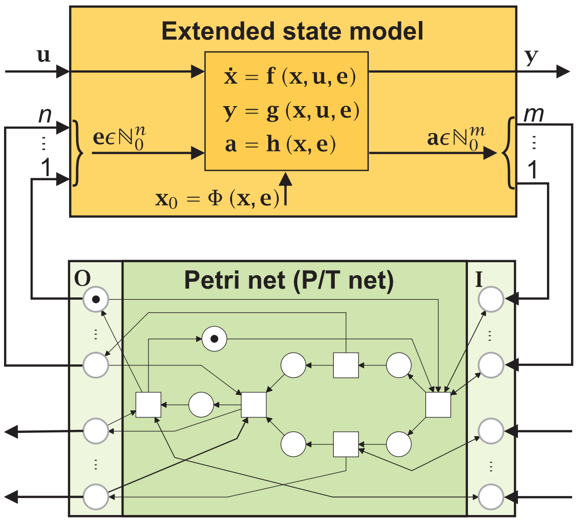
Extended state model
u
y
$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u},\mathbf{e})$
$\mathbf{y} = \mathbf{g}(\mathbf{x},\mathbf{u},\mathbf{e})$
$\mathbf{a} = \mathbf{h}(\mathbf{x},\mathbf{e})$
n
m
1
1
$\mathbf{e}\epsilon\mathbb{N}_0^n$
$\mathbf{a}\epsilon\mathbb{N}_0^m$
$\mathbf{x}_0 = \Phi(\mathbf{x},\mathbf{e})$
O
I
Petri net (P/T net)

Fig. 1. Hybrid model

troller by binary or nonnegative integer signals. Any continuous, hybrid or discrete partial model can be coupled with the controller, therefore modular and efficient modelling of large systems is possible. The internal dynamics of the discrete systems are modelled by P/T nets PN with the marking of the internal places M, extended by output places and additional input places (fig. 1) (Orth *et al.*, 2002).

To treat input signals provided by an input vector $\boldsymbol{i}$, the net is extended by the input place set I with a marking always representing the corresponding input signal. Input places are connected to transitions in the Petri net by read-only arcs since they represent signals as external firing conditions which cannot be changed. These arcs can be test arcs or inhibitor arcs, which allow or prevent the firing of the adjacent transition if the external signal's value is greater than or equal to the arc weight w. Internal places are connected with normal flow arcs only.

The output vector $\boldsymbol{o}$ consists of the marking of the output place set O, which is a subset of all internal places. Often input and output place capacities are set to one to reflect binarity of input and output signals. The model is comparable with signal interpreted Petri nets (SIPN) (Frey, 2002), but in particular the problem of contradictory, unspecified or redundant output is avoided by the concept of output places, while the occurrence of unstable markings (leading to cycles) and also indetermination is possible (Jörns, 1997).

The internal dynamics of the controller are represented by a P/T net which may be defined as 6-tuple $PN = (P, T, R, C, W, M_0)$. PN consists of the set of places $P = \{p_1, \ldots, p_i, \ldots, p_{|P|}\}$, the set of transitions $T = \{t_1, \ldots, t_i, \ldots, t_{|T|}\}$, the set of arcs (flow relation) $R \subseteq (P \times T) \cup (T \times P)$, the mapping $C : P \to \mathbb{N}$ defining the capacities of all places, the mapping $W : R \to \mathbb{N}$ giving the weight of every arc in R and the mapping $M_0 : P \to \mathbb{N}_0$ determining the initial marking of every place $(0 \leq M_0(p) \leq C(p) \forall p \epsilon P)$ with non-empty, finite and disjoint sets P and T (Abel, 1990).

Formally the event discrete subsystems are defined as P/T nets extended by inputs and outputs, resulting in the 12-tuple $PNIO = (P, T, R, C, W, M_0, I, O, R_{\mathrm{I}_{\mathrm{test}}}, R_{\mathrm{I}_{\mathrm{inhibit}}}, C_{\mathrm{I}}, W_{\mathrm{I}})$ with:

(1) the sets P, T, R, C, W, M_0 defined as above
(2) the set of input places $I = \{i_1, \ldots, i_i, \ldots, i_{|I|}\}$ with $I \cap (P \cup T) = \emptyset$,
(3) the set of output places $O = \{o_1, \ldots, o_i, \ldots, o_{|O|}\}$ with $O \subseteq P$,
(4) the input test relation $R_{\mathrm{I}_{\mathrm{test}}} \subseteq (I \times T)$
(5) the input inhibitor relation $R_{\mathrm{I}_{\mathrm{test}}} \subseteq (I \times T)$
(6) the mapping $C_{\mathrm{I}} : I \to \mathbb{N}$ giving the capacities of the input places,
(7) the mapping $W : (R_{\mathrm{I}_{\mathrm{test}}} \cup R_{\mathrm{I}_{\mathrm{inhibit}}}) \to \mathbb{N}$ giving the weight of the input read-only arcs.

In common Petri nets the moment of firing of an enabled transition is not determined. An enabled transition may fire but does not have to. To guarantee deterministic behaviour with respect to time for the discrete parts of the system, every transition of the net has to fire instantly as soon as it is enabled. The dynamic behaviour of the model is described by algorithm 1.

Algorithm 1 Behaviour algorithm

```
initialisation of net with initial marking $M_0$
initialisation of continuous part with $\boldsymbol{x}_0$
calculation of continuous output vector $\boldsymbol{a}$
loop
  repeat
    firing of maximal steps in discrete systems
    calculation of the marking $M$
    determination of discrete output vector $\boldsymbol{o}$
  until there are no activated transitions
  reinitialisation of the variable $\boldsymbol{x}$ if needed
  repeat
    integration of the differential equation
    determination of $\boldsymbol{f}$, $\boldsymbol{g}$ and $\boldsymbol{h}$
  until an element of $\boldsymbol{a}$ changes
end loop
```

Also, the implementation of the Petri net model on a controller will solve conflicts between multiple transitions by firing the transition with the lowest index. This solution allows the design of supervisory controllers which prevent unwanted or force favourable behaviour. Both aspects are indispensable for a logic controller which is expected to be deterministic. The duration of firing a single transition is assumed to be zero. After the firing of all enabled transitions in a maximal step, the changes in the discrete parts of the system result in an output vector $\boldsymbol{o}$ of the discrete subsystem.

To model the internal continuous dynamics of the continuous subsystems, switched differential equations are represented by an extended state model with continuous and discrete inputs. This state model allows the re-initialisation of continuous states and can be extended with depending functions depending on the continuous state (Müller, 2002). Both binary and continuous output signals can also be attached to other discrete or continuous systems. Emphasis of this article is on modelling discrete subsystems for the design of discrete controllers, so the modelling of continuous aspects of the process is not deepened here.

3. RAPID CONTROL PROTOTYPING

Typically, the design of an automation solution starts with a summation of the number of input and output interfaces. At this point, the need in computing power for the solution is also estimated and usually the hardware for the control realisation is chosen early. On the contrary the RCP method allows late determination of the target hardware, as the implementation of control algorithms with RCP tools usually gives great independence of manufacturer specific design tools.

3.1 Simulation

The RCP method consists mainly in the idea of starting with a simulation model of both process and automation. This model is improved during development so that implementation of the automation solution is done by generation of executable code from the automation algorithms for the target platform. The shift to graphically oriented environments allows for high efficiency and leads to less faults.

Apart from the final realisation as a 4^{th} aspect of RCP, there are 3 different types of simulation configurations during the development:

System Simulation: all components are simulated on development hardware, which is not necessarily the application target,

Software-in-the-Loop (SiL): the designed control algorithm is executed on development hardware, connected to components or the plant,

Hardware-in-the-Loop (HiL): the designed control algorithm is executed on the target hardware and attached to a simulation model of the process on development hardware.

3.2 Procedure Model

An important property of RCP is the requisite of a model, as it is a model based design method. The exactness of the model is augmented in every step of the development process. Every improvement in the automation algorithm model is transferred to the controller by code generation. This procedure usually demands for a single software tool for development or an integrated tool chain.

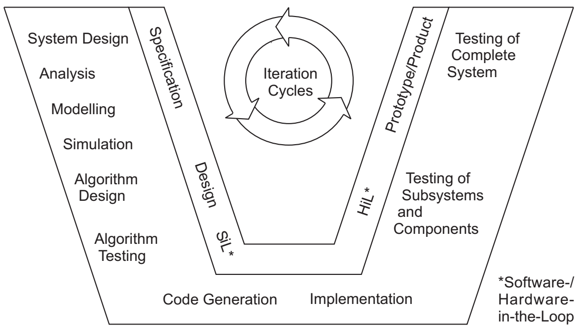

Fig. 2. V-Model

A possible procedure model for development processes allowing for iteration cycles is the V-Model (fig. 2). Starting at the upper left of the diagram with high abstraction like the problem specification, the detailing is augmented step by step going down the *V*'s left side, ending on a level of binary executable code. The right side leads again to an increase in abstraction by testing components, prototypes and finally the complete automation solution, which is in fact the last prototype produced by this process.

4. RAPID SEQUENTIAL CONTROL PROTOTYPING

In the following an example for a tool chain is presented for rapid prototyping of sequential control applications. The chosen model (sec. 2) was realised in the Petri net tool Netlab developed at IRT since the late 1980's (IRT, 2005), and though the approach to RCP is of general nature, notes specific to the chosen model, application and tool chain are made.

4.1 Tool chain

As Matlab/Simulink is a typical component of RCP tool chains in control design, this software package was chosen as basis which permits e.g. analysis, simulation and code generation for continuous systems. Stateflow is a well known toolbox for Simulink which allows the design of DES as statecharts, but these are not as common as Petri nets in industrial automation since SFCs (IEC EN DIN 61131-3, 1993) are closely related to them. Apart from that, Stateflow draws its power from complex textual annotations. A visual representation of concurrency and synchronisation is not allowed, so Petri nets are an important extension of Simulinks modelling possibilites. One way to enable simulation is a generation of Petri net code (subsection 4.3) that can be used during system simulation, SiL, HiL and also in the prototype.

Starting with a system simulation, the process model has to be replaced by an I/O interface to permit SiL, while the simulation has to run in soft or hard real-time. For this purpose a Profibus DP Remote-I/O can be accessed from a Simulink extension using TwinCat I/O (Beckhoff Automation, 2005). For HiL, the same interface can be used with reversed inputs and outputs. The realisation of the control protoype can be generated using the Real-Time Workshop (RTW), resulting in a function which has to be executed every control cycle of the Soft-PLC TwinCat.

A second example for an implemented RCP tool chain heads to the free process control platform ACPLT (PLT, 2005). The tool chain also consists of generating code with Netlab, Simulink and RTW. The RTW's code is then wrapped and compiled as a function block for ACPLT/FB for SiL and HiL (Orth *et al.*, 2004).

4.2 Analysis support

One aim of the proposed development environment for sequential control is to integrate analysis and synthesis methods into a RCP tool chain. An example for these features are online calculation and display of P- and T-invariants (Orth *et al.*, 2004). This enables e.g. easy checking of a synthesized supervisor (Moody and Antsaklis, 1998). Generation and display of reachability graphs is also supported by Netlab, and the use of a PNML document type definition (DTD) based on the DTD for P/T nets as data format allows for easy integration of other tools and interaction with them (Weber and Kindler, 2003).

4.3 Codegeneration

The generation of executable code is an important step to close the gap between simulation and controller implementation. To address Soft-PLC environments and a large variety of microprocessors, the C-language was chosen, even if unfortunately most development environments for contemporary hardware PLCs do not support the integration of other languages than specified by IEC 61131-3. Focussing on these PLC designs, the generation of instruction list code would be another possible target language (Frey *et al.*, 2001).

Automation hardware typically works cycle based and most controllers divide a cycle into three steps: after reading input signals, internal states are updated before output signals are written. If the controller consists of discrete event and continuous control algorithms, standard integration algorithms for the continuous part are provided by Simulink/RTW. Apart from the update of I/O, which is usually a question of configuration, the main task of the discrete control is executing the firing rule for the Petri net based algorithm. This means to ensure that all activated transitions are fired during the step and that a stable marking is reached. This marking will typically change only for changes in the Petri net inputs.

The execution of a single transition consists of the check for all activation conditions and of the firing of the transition if the transition condition is true, leading to changes in the marking of some or all places in the transition's preset and postset.

The transition condition is a boolean expression indicating if a transition is activated. Every activated transition is fired instantly (cf. section 2). The check of the transition condition can be devided in four parts:

- Check of all test arcs,
- Check of all inhibitor arcs,
- Check of all flow arcs for the preset,
- Check of all flow arcs for the postset, eventually with respect to the preset if self-loops are connected to the transition.

The transition action can be composed by execution of the flow arcs for preset and postset with special treatment of self-loops.

The resulting code may be executed in a loop until no transition is enabled. Optimisation of execution and sequence of the resulting code can be achieved using structural properties of the net, simulation results or reachability analysis (Frey, 2002; Girault and Valk, 2002). This is an important aspect of code generation for Petri nets but beyond the scope of this paper.

5. EXAMPLE OF A FLEXIBLE MANUFACTURING SYSTEM

Figure 3 shows a section of two machines in the upper and lower right, served by a transportation system which provides the machines with workpieces, which enter and leave the section at the left. Transportation possibilities for workpieces are given by the linear movement of the transportation system running on rails in the middle of the system, turning of turntables at the entry and the exit of each machine section and by using the conveyor belts. Conveyor belts are installed on the transportation system, the turntables and conveyor units. There are 30 binary signals controlling the different directions of movement, and 27 binary sensors indicating positions of turntables, transportation system, machines and workpieces.

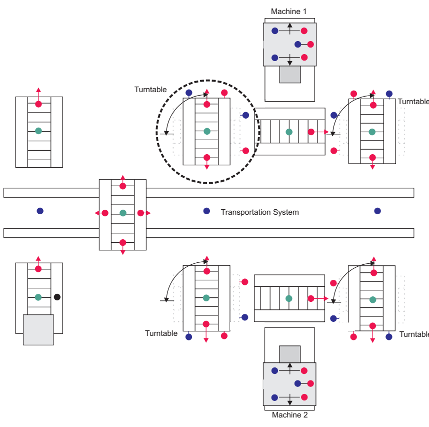

Fig. 3. Flexible manufacturing system scheme

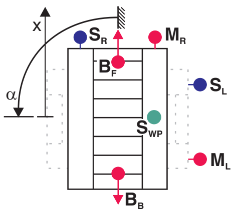

Fig. 4. Turntable scheme

In the following a single turntable with a conveyor belt is discussed, encircled in fig. 3. Two continuous states can be assigned to the turntable: the position angle of the turntable α (eq. 1) and the position of a workpiece x relative to the (rotated) axis in direction of the conveyor belt, if a workpiece is present on the conveyor belt (Marking M in fig. 5). The variable x is reset to $\mp L/2$ if a workpiece enters in the direction of the x-axis or against it respectively. If the rotational position α is outside of an allowed sector, errors will occur.

$$\dot{\alpha} = K_{\dot{\alpha}} \cdot \begin{bmatrix}+1 & -1\end{bmatrix} \begin{bmatrix}M_{\mathrm{L}} & M_{\mathrm{R}}\end{bmatrix}^T \tag{1}$$
$$\dot{x} = K_{\dot{x}} \cdot \begin{bmatrix}+1 & -1\end{bmatrix} \begin{bmatrix}B_{\mathrm{F}} & B_{\mathrm{B}}\end{bmatrix}^T \tag{2}$$

The turntable can be turned to the right by setting the motor signal $M_{\mathrm{R}} = 1$, to the left by $M_{\mathrm{L}} = 1$. The sensors S_{R} and S_{L} indicate if the turntable is in the right ($\alpha \approx 0°$, eq. 3) or left position respectively ($\alpha \approx 90°$, eq. 4). The sensor signal S_{WP} equals one if a workpiece is in the center of a conveyor unit (eq. 5). The conveyor belts are moved forward or backwards for $B_{\mathrm{F}} = 1$ or $B_{\mathrm{B}} = 1$. Situations where the contrary movement signals ($M_{\mathrm{R/L}}$, $B_{\mathrm{F/B}}$) are both equal to one at a time should be avoided.

$$S_{\mathrm{R}} = 1 \text{ iff } -\frac{B_\alpha}{2} \leq \alpha \leq \frac{B_\alpha}{2}, \quad \text{else } 0. \tag{3}$$
$$S_{\mathrm{L}} = 1 \text{ iff } 90° - \frac{B_\alpha}{2} \leq \alpha \leq 90° + \frac{B_\alpha}{2}, \quad \text{else } 0. \tag{4}$$
$$S_{\mathrm{WP}} = 1 \text{ iff } -\frac{B_x}{2} \leq x \leq \frac{B_x}{2}, \quad \text{else } 0. \tag{5}$$

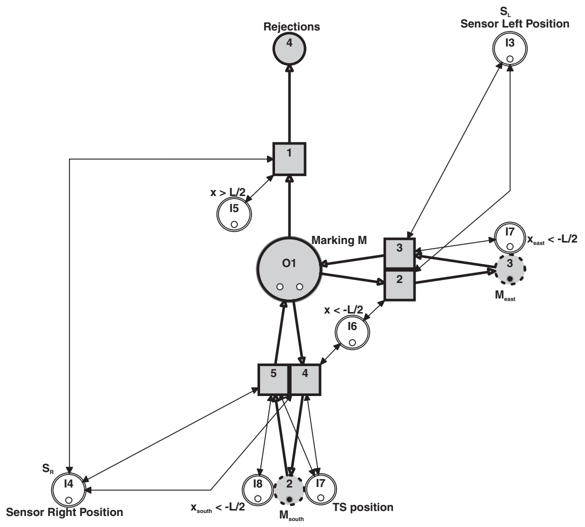

Fig. 5. Turntable model

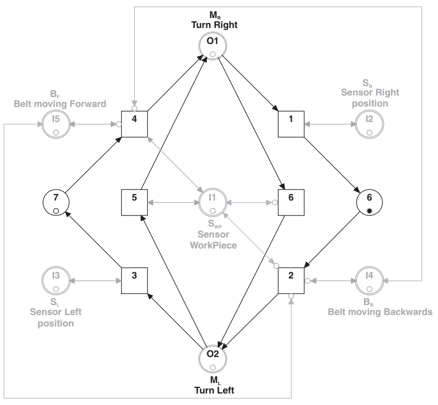

Fig. 6. Autonomous turntable rotation automation

The desired behaviour of the turntable in question is shown in fig. 5. Here workpieces may be transported from the transport system (in the 'south' of the turntable) to machine 1 (in the 'east') by firing transitions 5 and 2 and in opposite direction by firing transitions 3 and 4. Firing transition 1 allows transport off the turntable (in 'north' direction) in case of rejections. The net is clipped at the places encircled with dotted lines.

As a model of the uncontrolled turntable is needed for RCP, also the undesired behaviour has to be modelled. Several transitions activated for different error conditions with an error place in the postset were modelled separately. A central analysis aim is the proof that an error place is never marked for the controlled process. Verification model and results are too detailed for the general character of this article and have to be omitted.

The turntable rotation is controlled by the autonomous automation in fig. 6 which was tested with the process model before using it for SiL and further RCP simulations after generation of C-code. The following snippet of the code which checks the activation and eventually fires transition 2 gives an idea of the high readability of the generated code. Even if changes to the generated code are contradictory to the idea of a RCP tool chain, the possibility of changes and checking of the generated code is a necessity for acceptance of the method in practice.

```
// Code for TRANSITION_2
if
( // Check inhibitor edges
( !(max(min(INPUT_1_MARKING,INPUT_1_CAPACITY),0)
>= EDGE_23_WEIGHT) &&
!(max(min(INPUT_4_MARKING,INPUT_4_CAPACITY),0)
>= EDGE_28_WEIGHT) &&
!(max(min(INPUT_5_MARKING,INPUT_5_CAPACITY),0)
>= EDGE_30_WEIGHT)) &&
( // Check preset edges
PLACE_6_MARKING >= EDGE_3_WEIGHT) &&
( // Check postset edges
PLACE_4_MARKING + EDGE_4_WEIGHT
<= PLACE_4_CAPACITY)
)
{ // Changes by preset edges
PLACE_6_MARKING -= EDGE_3_WEIGHT;
// Changes by postset edges
PLACE_4_MARKING += EDGE_4_WEIGHT;
}
```

6. CONCLUSION

In this article a Petri net based model to describe technical processes as uncontrolled system combined with a discrete control is presented. The model involves modules of P/T nets, extended by inputs and outputs and read-only arcs for modelling programmable logic controller programs.

The general idea of rapid prototyping for control systems is introduced and applied to sequential process control, allowing for integration of usually unused analysis and synthesis methods into a development environment. An example realisation of a tool chain which allows Software- and Hardware-in-the-Loop simulations is presented as extension to a widespread simulation application.

Finally, the idea is demonstrated by an example for the prototyping of a flexible manufacturing system. The implementation of a control algorithm, using the chosen Petri net model and leading to the generation of executable code, is presented and discussed in detail.

REFERENCES

Abel, Dirk (1990). *Petri-Netze für Ingenieure*. Springer. Berlin.

Antsaklis, P. and X. D. Koutsoukos (1998). On Hybrid Control of Complex Systems: A Survey. In: *Proceedings Hybrid Dynamical Systems*. ADPM '98. Reims, France. pp. 1–8.

Beckhoff Automation (2005). TwinCat I/O. `http://www.beckhoffautomation.com`.

David, R. and H. Alla (1994). Petri Nets for Modeling of Dynamic Systems - A Survey. *Automatica* **30**(2), 175–202.

Frey, Georg (2002). Design and Formal Analysis of Petri Net Based Logic Control Algorithms. Dissertation. Universität Kaiserslautern, FB Elektro- und Informationstechnik.

Frey, Georg, Mark Minas and Karl-Heinz John (2001). Integration von Petrinetzen in den Steuerungsentwurf nach IEC 61131. In: *Proc. of the SPS/IPC/Drives 2001*. Nürnberg.

Girault, Claude and Rüdiger Valk (2002). *Petri Nets for System Engineering*. Springer. Berlin.

IEC EN DIN 61131-3 (1993). Programmable controllers - Part 3: Programming languages. International Standard.

IRT, Institute of Automatic Control, RWTH Aachen University (2005). Netlab. `http://www.irt.rwth-aachen.de/typo3/index.php/Petrinetz-Tool_Netlab/101/0/`.

Jörns, Carsten (1997). Ein integriertes Steuerungsentwurfs- und Verifikationskonzept mit Hilfe interpretierter Petri-Netze. Dissertation. Universität Kaiserslautern, FB Elektro- und Informationstechnik.

Lunze, J. (2002). *Automatisierungstechnik*. Vorlesungsskript. Bochum.

Moody, John O. and Panos J. Antsaklis (1998). *Supervisory Control of Discrete Event Systems Using Petri Nets*. Kluwer Academic.

Müller, Christian (2002). *Analyse und Synthese diskreter Steuerungen hybrider Systeme mit Petri-Netz-Zustandsraummodellen*. Vol. 930 of *Fortschritt-Berichte VDI Reihe 8*. VDI-Verlag. Düsseldorf.

Orth, Ph., Ch. Müller, D. Abel and H. Rake (2002). Synthesis of a discrete control for hybrid systems by means of a petri-net-state-model. In: *Modelling, Analysis, and Design of Hybrid Systems* (S. Engell, G. Frehse and E. Schnieder, Eds.). Springer. Berlin.

Orth, Philipp, Alexander Bollig and Dirk Abel (2004). Rapid Control Prototyping diskreter Steuerungen in der Automatisierungstechnik. In: *Proc. of the SPS/IPC/Drives 2004*. Nürnberg.

PLT, Chair of Process Control Engineering, RWTH Aachen University (2005). ACPLT. `http://acplt.plt.rwth-aachen.de/fb/fb.html`.

Weber, Michael and Ekkard Kindler (2003). The Petri Net Markup Language. In: *Petri Net Technology for Communication Based Systems* (H. Ehrig, W. Reisig, G. Rozenberg and H. Weber, Eds.). Vol. 2472 of *LNCS*. Springer. Berlin. pp. 124–144.