Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

HPNS – a Hybrid Process Net Simulation Environment Executing Online Dynamic Models of Industrial Manufacturing Systems

Sebastian Bohlmann

Department of Simulation and Modelling
Leibniz University Hannover
Welfengarten 1, 30167 Hannover, Germany

Volkhard Klinger

Department of Embedded Systems
FHDW Hannover
Freundallee 15, 30173 Hannover, Germany

Helena Szczerbicka

Department of Simulation and Modelling
Leibniz University Hannover
Welfengarten 1, 30167 Hannover, Germany

ABSTRACT

Modelling technical systems nowadays is a great challenge of automation technology. Particular complex manufacturing processes, like the industrial paper production, consists of discrete and continuous signals and dynamic response times. Inspired by Petri nets, this paper proposes a new modelling approach is presented to describe those technical systems on different abstraction layer. The hybrid process net simulator (HPNS) framework allows to specify complex models, typically represented as a system of differential equations mixed with continuous and discrete-event subsystems, based on a bipartite graph structure. In addition, the HPNS execute model specifications with dynamic delays.

This paper focuses on the concept and the modelling approach. We give a short review of this new unified representation model for hybrid technical systems, the presentation of the model formalism is out of scope of this paper. A short summary about capabilities and restrictions of HPNS is presented. Moreover, examples of hybrid systems are presented.

1 INTRODUCTION

Technical systems are characterized by a variety of different components which imply only a part-specific and heterogeneous description due to the mechanical, electronic and information-processing subsystems. Hence, the modelling of the entire system in a closed simulation framework is usually not possible. Hardware/software-cosimulation represents a good example for this situation: Based on an interface and a synchronization process the simulators for hardware and software subsystems are coupled without any standardized model view. In particular the automation technology provides no closed model-based simulation methods for both, neither the process nor the necessary process control.

In this paper we show a strategy to describe complex technical systems consisting of

- Physical processes,
- Chemical processes,
- Mechanical systems,
- Hardware-subsystems,
- Software-subsystems

in a closed model-based simulation environment. Physical, chemical and mechanical subsystems are characterized by continuous behaviour. Hardware subsystems can be characterized by continuous and discrete behaviour, while software is characterized by discrete behaviour.

978-1-4244-5771-7/09/$26.00 ©2009 IEEE

Before introducing one particular example of such a complex system – the industrial paper manufacturing – in subsection 1.1, we list the boundary conditions for such a simulation framework, called hybrid process net simulation framework (HPNS).

- The simulation framework is able to handle subsystems described by differential equations.
- The given technical process contains a flow-related behaviour and discrete events.
- Different subsystems are connected by feedback loops and other relations.
- The given technical process consists of continuous and discrete signals.
- Static and dynamic delays and response times are possible.
- The given technical process is very complex; it is composed of thousands of signals, relations between process subsystems and process states.
- Two different operations-modes have to be supported:
  - Offline-analysis
    This type of analysis help to understand the process behaviour based on given process data localised in an archive database. It can be used to rerun process transitions and to replicate certain process states.
  - Online-analysis
    The HPNS framework is integrated into the technical process to evaluate and to analyze the online available process data. The objectives, like software-sensoring, process supervision and model-based prediction require an efficient simulation environment capable of being integrated into the process.

According to these very abstract boundary conditions we can discuss process-related aspects to precise them and to deduce some additional constraints. We have to analyze later on in subsection 1.2 which requirements have to be taken into account in this paper and which ones are fulfilled using simplifying assumptions with regard to the given examples in section 5.

Obviously for modelling we imply a hybrid approach, where hybrid means the support of continuous and discrete signals. Furthermore we need a process model to specify the given process in a standardized and formal description. This model forms the technical process specification using a new model approach, presented in section 4. Based on a model simulator an executable process model is available for the online- and offline-analysis. Later on we discuss the whole HPNS framework and the design process from the technical process up to the executable model. Before refining the requirements we have to introduce the technical process under consideration, the industrial paper manufacturing. This technical process is the reference process in the project context where this paper is elaborated in. The model and simulation framework have been developed to meet all different requirements of this process.

1.1 Specific Problems in Industrial Paper Manufacturing

The pulp and paper industry as one instance for employment is a qualified process according to our requirements listed above (Viitamäki 2004). There are thousands of continuous and discrete signals describing the behaviour and states of the different subsystems. Hundreds of control loops and relations are existing in between the subsystems. The dimension of a production line, which typically has a length of typically more than 200m, is difficult to handle. There are many backpropagations of state changes and there is almost no separation between the different process steps. Paper manufacturing is in fact a continuous process with high speed behaviour (production speed up to 35m/s), requiring high performance computation for process control (Ast 2007). Numerous quality parameters used for control purposes can not be measured at the position on which the control authority exists. This leads to very high response times in process control loops and is complicating the stable process control at all.

For this process no detailed model exists enabling for example an evaluation of the process quality. The manufacturing steps had been developed by empirically over decades. Therefore up to now techniques being able to optimize the manufacturing process at all, are scarcely or not used. Two of these techniques are soft sensors and model predictive control. Whereas soft sensors are able to refine the process understanding by a derivation of so called inner process states, the model predictive control provides a set horizon N_u in which the change of process signals and derived quality parameters are foreseen within a future period $[t_k \dots t_{k+Nu}]$.

1.2 Requirements and Objectives

Based on the above boundary conditions and the reference process presented in subsection 1.1 we now can refine the requirements for the HPNS framework (Bohlmann and Klinger 2007).

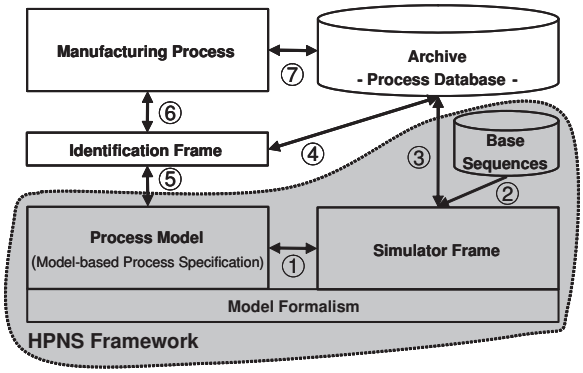

Figure 1: HPNS architecture

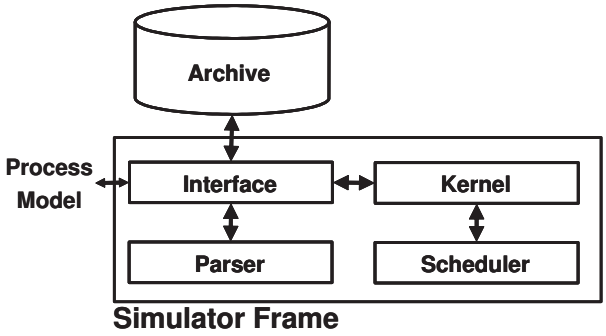

Figure 2: Simulator Frame

R_1: HPNS is able to handle subsystems described by differential equations. All related processes contain mechanical subsystems have to be described in such a way.

R_2: HPNS supports continuous and discrete signals. In greater detail HPNS supports value- and time-discrete and value- and time-continuous signals.

R_3: HPNS allows any process topology therefore all possible feedback and control loops. All possible arithmetic and logical relation between different process elements are valid in HPNS.

R_4: HPNS supports static and dynamic delays (response times).

R_5: HPNS is optimized for high performance simulation (Fishwick 1994). This requirement is not taken into account in the scope of this paper.

R_6: HPNS supports two different modes of operation: online- and offline-analysis.

Fitting all these requirements the HPNS framework has the key objective to realize a process model execution for two operation modes, offline- and online-analysis. Therefore it can be used for several embedded automation scenarios, for example software-sensoring and model-predictive control.

In the next sections we introduce the HPNS architecture.

2 HPNS ARCHITECTURE

In this section we present the HPNS architecture, depicted in Figure 1. The entities, their characteristics and relations are described briefly in the following list. Additionally we introduce some terms used in this paper.

- Model Formalism
  The model approach covers several formalism describing the subclasses of the given system, like discrete subsystems, continuous subsystems, etc.. Moreover, it allows to integrate all different subsystems into one formalism. This is necessary because it is not possible to fit many technical phenomena into one single formalism, like classical Petri nets. The special formalism describes its elements and their relations based on mathematical definitions. The process model and the simulator frame are defined according to this formalism. We introduce the model approach and its characteristics in section 4.
- Process model
  The process model describes the technical or manufacturing process by means of the model formalism. It corresponds to the system specification and takes all subsystems and interfaces of the technical process into consideration. The abstraction level of the specification is oriented on the knowledge of the technical process and the required accuracy. It can be available in various abstraction levels. The modelling process transforms a technical process into a process model that is accomplished using the identification process, localised in the identification frame. The bilateral relation 5 in Figure 1 shows the dependency. The process model is generated by the identification frame which is executing an ongoing optimizing during simulation.

- Simulator Frame
  The simulator frame consists of four modules, presented in Figure 2 (Banks et al. 2000; Zeigler, Praehofer, and Kim 2000; Law and Kelton 2006):
  - Interface
    The interface manages the communication between the process model, the archive and the base sequences (relation 1, 2 and 3 in Figure 1 clarify the information flow). Depending on different operation modes, like online-analysis, offline-analysis and process supervision, different communication modules can be linked to the simulator execution. The following extendable communication links are implemented:
    * Base sequence input form XML (extensible markup language) definition file.
    * Database module for base sequence input or simulation output. Eclipse BIRT (business intelligence and reporting tools) can be used to provide high quality reports, because the database server used has a ODA (open data access) interface.
    * OPC-DA (ole for process controls-data access) DCOM (distributed common object model) interface to connect standard industrial automation equipment.
    * File in-/output for comma separated value text file with different formats and scalings.
    * RPC (remote procedure call) network wrapper for all other communication modules.
  - Parser
    This component is capable to instantiate and configure all modules and links between them. Two interfaces are provided for this purpose. First a XML-File parser, which is primary used for testing and to describe online experiments. Second a API for direct interactions with a simulation run and construction or modification of models. It is mapped via RPC technique over a network socket to be able to connect to a simulation server witch is embedded into some process. In addition a graphical editor to interact by this API is available.
  - Kernel
    The simulation kernel is used to execute a model previously defined. All modules are written in pure java without special libraries (except the standard JDK). Using its internal object based representation it generates a discrete event simulation. For each type of event code fragments are dynamically generated, compiled and loaded into the virtual machine. This method is used to counterbalance the negative performance effects from the complex modelling formalism (Abel and Bollig 2006). It speeds up the simulation by elimination of unused parts. Especially all functions used by the model can run in native code without interpretation. With regard to requirement R_2 time continuity is realized by an hierarchical time model using delta cycles. These delta cycles consume no time in simulation (wall clock time) but time for simulation: This hierarchical time model is used in well known simulators like SPICE or VHDL.
  - Scheduler
    The central scheduler component however is static over time. Each event is stores in the heap and ordered by three hierarchical layers. The highest is preserved for external communication events. For example wallclock time synchronisation or logging. The next one is used for wallclock time scheduling. Short time events (based on the configuration) in this layer are stored in a fixed accelerating memory structure. The lowest level provides a ranking based on dependencies between scheduled event types. All independent events in the lowest layer could be executed in parallel fashion on a multicore machine.
- Identification Frame
  The HPNS framework requires a process model as described above. The identification frame realizes a model design flow, based on the archive (Relation 4 in Figure 1 the dependency respecting the archive (historical data) and based on the running manufacturing process (relation 6 in Figure 1 shows the dependency respecting the current data). Two different operation modes generating the process model are distinguished:
  - Pre-modelling
    The model is build up step-by-step before the realization of a technical process to optimize the realization parameters.
  - Post-modelling
    An existing process will be modeled based on the available process data (archive data). The generating procedure consists of several preprocessor steps executing a complex data processing.

Both operation modes are completed by an optimizing procedure evaluating the correspondance in between the process model, the archive and the technical process. The identification frame is not in the scope of this paper.

- Manufacturing Process
  The manufacturing or in more general the technical process is given by the application, in this project for example the industrial paper manufacturing. The process data are archived in the process database (relation 7 in Figure 1).

As shown in Figure 1, the HPNS framework consists of the entities Process Model, Simulator Frame, Model Formalism and Base Sequences. The other modules are evaluated and tested during ongoing research and will be published soon.

In the next sections we will first introduce some foundations for modelling.

3 MODEL FOUNDATIONS

To fit the requirements and objectives introduced in section 1 we have to decide for a method or strategy concerning modelling and simulation. Several tools and frameworks exist, like Matlab/Simulink, Statemate, Speedchart, COSSAP, SPICE, MIMOLA, etc.. But all these approaches are optimized for a particular type of application, for example for the design of digital signal processors from a very high-level behavioral specification (MIMOLA) or the rapid prototyping of systems via Simulink. All these tools and frameworks are not fitting all the given requirements R_1 - R_6.

Our approach to develop a model-based simulation was inspired by the theory of Petri nets. Petri nets allow formal and graph-oriented description for system modelling. Systems can be formed by concurrent processes and therefore they extend the automata theory. Petri nets are widely used to model discrete event dynamic systems, like communication protocols. Using continuous Petri nets, the markings of places are real numbers; the transition firings are continuous. They were defined more recently to model a continuous system. Most technical systems and of course manufacturing systems are not exclusively discrete nor continuous, but they are combined from discrete and continuous components. Hybrid Petri nets allow modelling of such systems. In the classical model developped by David and Alla (1994), the discrete part of HPN enables to model delays with threshold and the continuous part of HPN models the continuous flows without any thresholds and delays. A large number of extensions of Petri nets, like timed, continuous and hybrid, have been introduced with their characteristic features in David and Alla (2005).

We have evaluated several types of Petri nets to prove if there are able to fulfill the requirements. Considered exemplary we note some of these variants. The constant speed continuous Petri nets (CCPN) are obtained from discrete Petri nets by the fluidification of the markings. They combine continuous and discrete behaviour. Moreover the fluid stochastic Petri net (FSPN, (Trivedi and Kulkarni 1993; Tuffin, Chen, and Trivedi 2001)) are existing where the places can hold fluid rather than discrete tokens. A third type is represented by the Timed hybrid Petri nets (THPN)

These type of Petri nets offers a homogeneous description of hybrid systems consisting both, continuous and discrete signals. The hybrid extension refers to the places as well as to the transitions. Besides the continuous places and transitions the classical discrete places and transitions also exist in the description and are integrated into the net formalism. In Tuffin, Chen, and Trivedi (2001) is shown, that the FSPNs can be expressed as Hybrid Systems (Varaiya 1999). The modelling power of these variants is growing, moreover there are some more characteristics we have to take into consideration: Static and dynamic delays and/or response times (R 4). First we discuss static delay/response time scenarios to evaluate the well-known Petri net concepts. Obviously we focus on the hybrid Petri nets and their extensions.

An example system with delay on the continuous product flow is a conveyor or a normal tubbing. Since there is no threshold of the product volume, the delay in continuous product flow cannot be properly modeled by timed hybrid Petri Nets (THPNs). Several authors have purposed extensions of the original THPN-model in order to represent delays in the continuous flows. In David and Caramihai (2000) some features are added in addition to ordinary hybrid Petri nets (inhibitor arc, 0+ weight of an arc joining a continuous place and discrete transition, 0+ marking of continuous place). These are essential features describing preliminary continuous places and discrete events, even in the starting time 0. Another extension of hybrid Petri nets are the generalized batches Petri nets (GBPNs), introduced in Demongodin (2001). Based on the definition of batch places and batch transitions it is possible to model delays on continuous flows. Behaviour is described by the batch function, which is given by the speed of transfer, the maximal density and the length of place. In each batch place the set of input transitions consists of only one batch transition. In addition the set of output transitions consists of only one batch transition as well. These hybrid Petri nets extensions are characterized by similar expressive power with respect to modelling of continuous flows. But when the dynamic delays/response times are considered it is obvious that any approporiate methodology does not exist. The definition of GBPN allows variable delays, but only within a limited mode. It is not possible that an internal coherent batch overpass another one, as the variability of delay is restricted to distinguished points in time. In Demongodin and Koussoulas (1998) differential hybrid Petri nets (DHPN) are introduced composed of

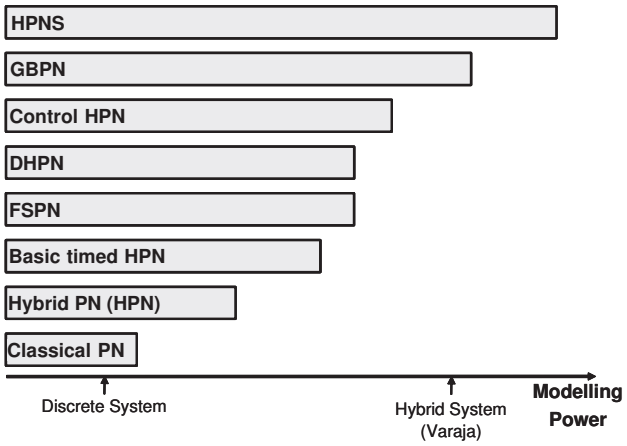

Figure 3: Modelling Hierarchie

two kinds of places and two kinds of transitions: discrete place and discrete transition, differential place and differential transition. They are powerful enough to model efficiently a hybrid system if the continuous system can be represented by n linear first order difference state equations. Therefore this Petri net variant can be arranged near to the FSPN; it provides a similar modelling power but another modelling scope. With regard to David and Alla (2005) our HPNS can be viewed as an extension to a control hybrid Petri net (ControlHPN).

This ControlHPN consists of discrete- and continuous-transitions, synchronized on events or conditions. For the discrete-transitions a timing is defined (dependency: stochastic, functions of time, marking, firing speeds), and also for the continuous-transition the flow rates are variable (dependency: functions of time, marking, firing speeds). In Figure 3 different variants of Petri nets are arranged with regard to their qualitative modelling power. The axis of abscissae is divided into sections due to the characteristics of system properties showing the increasing modelling power from discrete systems to hybrid systems and beyond where the complex manufacturing systems are localised.

Moreover the HPNS formalism extends definitions of discussed Petri nets variants with the following characteristics:

C_1 The HPNS-Model uses a special model for time.//] We define the time as a combination of a continuous part and a natural order of events at a time index to model an internal and external progress in time. The time index describes the time elapsed in the model.

$$\mathbb{T} : \mathbb{R} \times \mathbb{N}$$

C_2 The model is capable to express mixed dynamic discrete and continuous delay functions over time.
Of course a non negative delay is used to guarantee causality. On the other hand nondeterministic and parallel processes as one special strength of petri nets are still easy to implement.

C_3 Decision memory
The HPNS is not working like an finite state automaton or sequential logic system but takes a time background horizon into consideration to assign the next internal and/or external state. This time background horizon depends on the transition-specific delay to guarantee the correct superposition of discrete and continuous actions.

C_4 Interaction
HPNS considers an interaction between discrete and continuous signals not only at fixed time points due to the continuous character of interference.

C_5 Differential algebraic equations (DAE) as a representation for most physical processes are directly implied.

We introduce the HPNS model approch in the following section.

4 MODELLING APPROACH

In this section we give a short description how to model a complex system using HPNS-Framework. For the better comprehension we describe the modelling paradigm in an informal way. The model is inspired by classical petri nets (section 3) and several known extensions. It is primary designed to be used for simulation and data driven semi-automated model generation.

The basic model simulated by the HPNS-Framework is called process net (PNet). The topology of a PNet consists of a bipartite graph of transitions and places. Each has two subtypes:

- Transitions
  - C-Transition : Transition with primary continuous behaviour
  - D-Transition : Transition with discrete behaviour
- Places
  - I-Place : Place with integration characteristics over time
  - N-Place : Stateless place without integration

The corresponding symbols are shown in Figure 4. The different nodes are composed with some constrains (listed at the end of this section) with regard to the bipartite structure. In a process net several properties are connected with the edges of the graph. This is done likewise arc weights in P/T-Nets. The valid properties are depending only on the target of the edge. Edges with a transitions as target are uniquely named related to that transition. All places in a preset of a transition define a partial marking of the net used later.

4.1 Places

The N-Place is the simplest object in the net. Its marking at a time (see C_1 in section 3) is only defined by the current flow output of all transitions in the preset. It has only two properties. These are a constant minimum and maximum value. If one of the two borders is exceeded the value is simply truncated to that value. The marking is defined as:

$$\mathbb{M} : \mathbb{R} \times \mathbb{R}^{\alpha}$$

The addition of the scalar value and the first component of the vector is similar to the classical known scalar marking at one time index. The vector contains a finite number $(\alpha - 1)$ of derivations of the first component. The length of the vector markings is a constant value for one process net and is part of the initial formal definition.

For an I-Place the definition of the marking is identical to that of a N-Place. The only real difference in-between is the current marking over time is defined by all outgoing and incoming flows of the pre- and postset integrated over time. For this

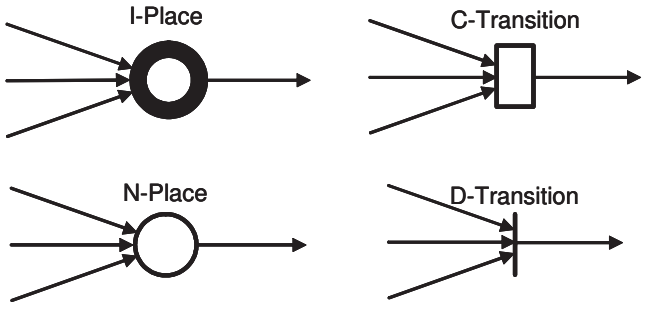

Figure 4: Process net symbols

purpose we order all tuples representing the time. By the help of this construct time can be reduced to a classical continuous and scalar time. Integrating continuous flows of C-Transitions with limited bandwidth is simple. Discrete behaviours in the net are perceived as dirac delta distributions or heaviside step functions, depending on the structure of the preset. Analogous to the N-Place two limiting properties can be linked with an I-Place.

4.2 Transitions

D-Transitions and C-Transitions have almost the same properties:

- Output Function
- Activation Function (non negative)
- Properties of incoming arcs:
  - Dynamic Delay Function (non negative)
  - Input Function

In addition the D-Transition defines an activation delay function. All functions use as domain of definition the partial marking of the transition mentioned before. So we start with the D-Transition. The input and output functions represent the weights of the transition; the values are real numbers. As the classical P/T-Net-Transition the D-Transition in process nets has a value-discrete behaviour. Because it could be activated at any time the behaviour over time can be continuous if the additional activation delay function is zero. The activation delay function defines the time the D-Transition is disabled after activation. It is re-enabled if the time elapsed since last activation exceeds the value of the activation delay function. Stochastic timings in this case are not directly supported at the moment. They can be modelled indirectly by the use of two extra nodes (D-Transition and I-Place) and the use of a random function provided by the simulator. If the activation delay is different from zero value changes become discrete in time. The delay function of arcs does not influence the targeted transition in a direct manner. Instead components of the partial marking are delayed. Delayed in this context means that values of historical markings are used for that arc. The time index of the used values is $t_c - \delta(a)$, with t_c is the current time and $\delta(a_t)$ the calculated delay. a_t represents the value of the delay function at time index t. The definition of the dynamic delay is compatible with a classical constant delay. It will result in the same constant value. In a simplified form it can be written in implicit form:

$$\delta(a) : \int_{\delta}^{t} \frac{1}{a_\tau} \mathrm{d}\tau = 1, a_t \neq 0$$
$$\delta(a) = 0, a_t = 0$$

As a direct consequence the state of a PNet can not be defined only by the actual marking. It is defined by the markings over time in the interval ending at the current timestamp t_c. The width of the interval is the maximum delay in the PNet at time t_c. This traces back to the fact that all delay functions have to be non negative, because otherwise causality is lost. The activation function is a simple guard function. If its result is false, then the transition is disabled.

To get an description of the C-Transition first of all the input/output functions have to be interpreted as functions defining the flow. The behaviour of the C-Transition depends on the behaviour of its partial marking. If this marking has only continuous behaviour, the output flow in general will be continuous too. Moreover, there are two main exceptions from this rule. First if there is an activation function, then the flow can be interrupted in a time-discrete manner. Second the output function could have a value-discrete or time-discrete characteristic. All the other properties of a C-Transition are portable from the description of a D-Transition.

The PNets are designed to be non autonomous. External events or signals can be linked with so called input places. If an external event is processed by the simulation the value change will happen immediately. No extra properties are provided by the HPNS-Framework.

With this knowledge we can define some structural constraints:

- If there is a D-Transition with a N-Place in its preset, then the input function of the corresponding arc is zero.
- External events or signals can only be connected with an N-Place.
- The known history of the external influences should be at least the maximum delay at the simulation start time.
- All external signals have to be bandwidth limited and the limit is known to the simulator.
- External events can be ordered in a lexicographical fashion and have a fixed scalar timestamp.

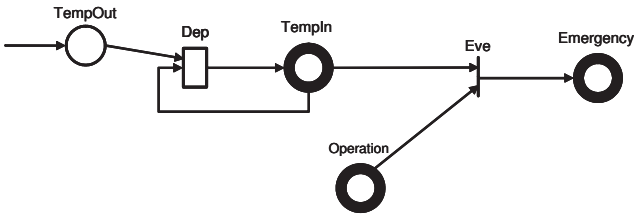

Figure 5: PNet from the simple temperature gradient

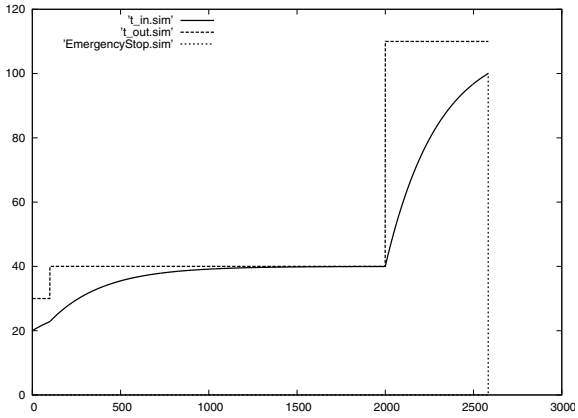

Figure 6: Inner and outer temperature

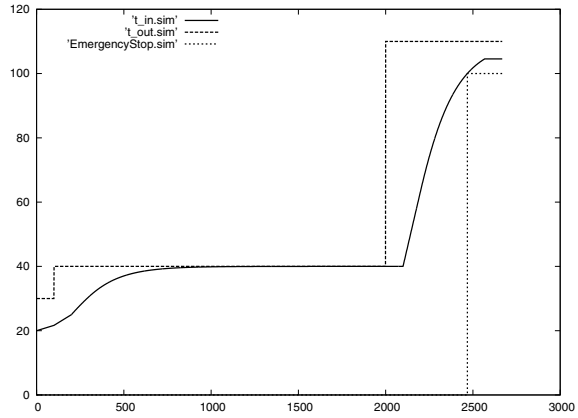

Figure 7: Inner and outer temperature with static delay

5 MODELLING EXAMPLES

In the following subsections we present different PNets simulated on HPNS. All results are shown using the transient behaviour of all model signals. The axis of abscissae represents the time elapsed marked by the ongoing time indexes.

5.1 Example: Simple Temperature Gradient

In this example two different spaces are given. They are separated by a border with a defined borderline. One space is the outer space, the other one is the inner space. If the temperature in the outer space is changed, the temperature in the inner space is accommodated due to a time constant. If the inner temperature is increasing over $100°C$, a discrete emergency stop is triggered. The PNet modelling this behaviour is shown in Figure 5. The central function of this net is handled in the C-transition 'Dep' which realises a feedback loop to derive the inner temperature (TempIn) depending on the outer temperature (TempOut) and their temperature gradient. The output of this C-transition is derived by:

$$\frac{\text{TempOut} - \text{TempIn}}{300}$$

This output is integrated in the I-place 'TempIn' representing the inner temperature. The I-Places 'Operation' and 'Emergency' describe the standby of operation state and the warning signal (event) coming up when the inner temperature is increased above $100°C$.

The results for the outer and the inner temperature (t_{in}, t_{out}) are given in Figure 6. The constant delay, which has no physical correspondence, can be modelled very simple by adding a delay to the C-transition, is shown in Figure 7. It can be seen between time index 2000 and 2100.

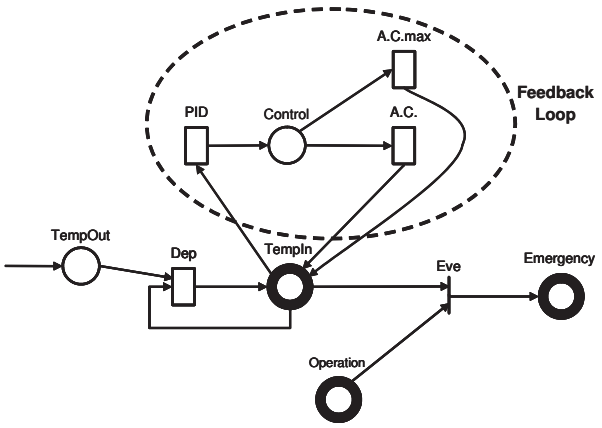

Figure 8: PNet of the temperature compensation

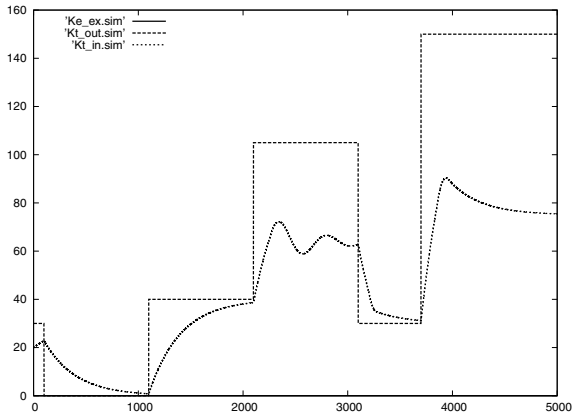

Figure 9: Simulation of the temperature compensation PNet

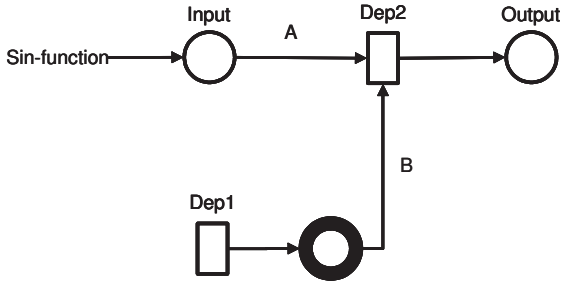

Figure 10: PNet of a dynamic delayed external signal (sinus)

5.2 Temperature Compensation with Air Conditioning

A more complex example is given in the PNet in Figure 8 which is based on the PNet in Figure 5 extended by a proportional plus integral plus derivative controller (PID, feedback loop). Based on this controller the temperature scenario is extended by an air conditioning unit, which controls the inner temperature (Figure 9). If the outer temperature is increasing over $60°C$, the air conditioning is working based on a closed-loop control (overshooting between time index 2400 and time index 3000 in Figure 9). Between time index 3100 and time index 3200 the delayed response of the control can be seen very good. After 100 time units, the closed-loop control decrease the cooling power because of the lower outer temperature.

5.3 Example: Dynamic Delay

In this example (Figure 10) we demonstrate the ability of the HPNS to calculate dynamic delay functions. The synthetic example uses a linear decreasing dynamic delay, represented by the I-Place in Figure 10. The external supplied input signal is delayed by the marking of that place. As external input signal we use a sinus function over time. Thus the bandwidth of the signal is easy to calculate. The resulting traces are shown in Figure 11. Figure 12 is a zoom of the interesting section when the output becomes valid. At the beginning the delay is high, decreasing continuous over time to zero. The output trace is undefined at the beginning in this simulation, because no information is provided before time index 0. The HPNS is starting the time where enough historical information is present. In fact a signal processing unit at the input of external signals, used to form and provide the necessary data for the kernel, adds a small delay too.

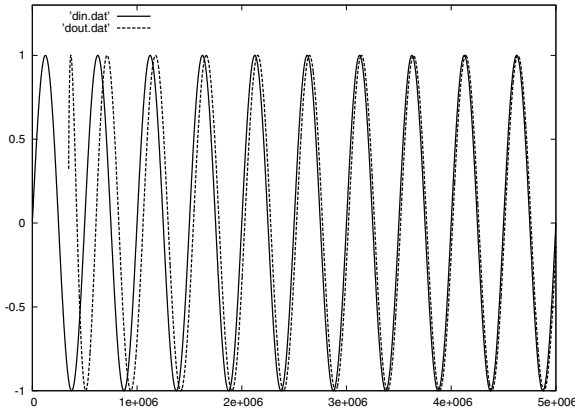

Figure 11: Trace of delayed external signal

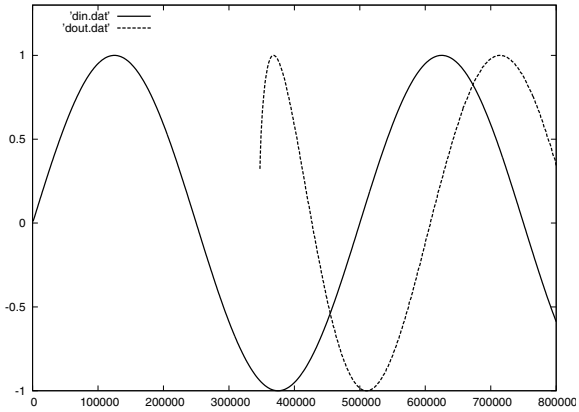

Figure 12: Zoom of figure 11

6 CONCLUSION

The objective of this publication was the establishment of a new modelling approach that can provide a unified representation for technical systems within a simulation framework. This model approach, inspired by hybrid Petri nets, covers several formalism describing the subclasses of the given system, like discrete subsystems, continuous subsystems and subsystems described by differential equations. It is integrated into the HPNS framework to support the two modes of operation, the online- and the offline-analysis, in an efficient way. Moreover the HPNS framework, based on the modelling approach, supports static and dynamic response times, an extension of the state-of-the-art variants of Petri nets, and combines most parts of these description formalism in one single method of modelling.

The particular characteristics are reached by several innovations, like the decision memory, interaction and superposition strategies. The decision memory expands the finite state character by a time background horizon, while the interaction and superposition strategy allows the HPNS framework an interaction between discrete and continuous signals not only at fixed time points due to the continuous character of interference. By the limitation of bandwidth of external interaction it is possible to connect the HPNS-Framework to existing automation systems. The contribution of the new model approach to the study of technical systems can be manifold. The model facilitates the description of complex technical processes using the powerful symbols. Based on the two operation-modes, the offline- and online-analysis of technical processes is simplified. Thus, the automation and the design of automation systems can be put on a new basis.

A semi-automated identification frame presented in Figure 1 is under development. Based on particular machine learning algorithms and certain characteristics (C_1 to C_5) of the HPNS model approach, it will provide an efficient data driven process identification.

ACKNOWLEDGMENTS

This work was supported by DREWSEN Spezialpapiere Gmbh & Co. KG.

REFERENCES

Abel, D., and A. Bollig. 2006. *Rapid Control Prototyping: Methoden und Anwendungen*. Berlin: Springer Verlag.

Ast, O. 2007. Diplomarbeit: Analyse und Projektierung der Automation eines Querschneiders in der Papierindustrie zur Optimierung der Schnitttoleranzen.

Banks, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. 2000. *Discrete-event system simulation*. 3rd ed. Upper Saddle River, New Jersey: Prentice-Hall, Inc.

Bohlmann, S., and V. Klinger. 2007. Modellbildung für kontinuierliche Produktionsprozesse in der Papierindustrie. *Forschungsberichte der FHDW Hannover* 08:1–20.

David, R., and H. Alla. 1994. Petri nets for modeling of dynamic systems: A survey. *Automatica, 30(2)*:175–202. InternalNote: Submitted by: khm@daimi.au.dk.

David, R., and H. Alla. 2005. *Discrete, continuous, and hybrid petri nets*. Berlin: Springer-Verlag.

David, R., and S. Caramihai. 2000. Modeling of delays on continuous flows thanks to extended hybrid petri nets. In *The 4th International conference on Automation of Mixed Processes: Hybrid Dynamic systems*, 343–350.

Demongodin, I. 2001. Generalized batches pns. *Discrete event dynamic systems: Theory and applications* 11.

Demongodin, I., and N. T. Koussoulas. 1998. Modelling of hybrid control systems using petri nets. In *Proceedings of the 3 rd International Conference on Automatisation des Processus Mixtes: les Systemes Dynamiques Hybrides*.

Fishwick, P. A. 1994. Computer simulation: Growth through extension. *Computer and Information Science Dept., University of Florida, CSE 301*:3–20.

Law, A. M., and W. D. Kelton. 2006. *Simulation Modeling and Analysis*. McGraw-Hill.

Trivedi, K. S., and V. Kulkarni. 1993. Fspns: Fluid stochastic petri nets. In *Lecture Notes in Computer Science; Proc. 14th International Conference on Applications and Theory of Petri Nets*, ed. M. A. Marsan, 691: 24–31, Heidelberg, Springer-Verlag.

Tuffin, B., D. Chen, and K. S. Trivedi. 2001. Comparison of hybrid systems and fluid stochastic petri nets. *Discrete Event Dynamic Systems: Theory and Applications* 11(1/2):77–95.

Varaiya, P. 1999. Design, simulation, and implementation of hybrid systems. In *Application and Theory of Petri Nets 1999, 20th International Conference, ICATPN '99*, ed. S. Donatelli and H. C. M. Kleijn, 1–5. Williamsburg, Virginia, USA.

Viitamäki, P. 2004. *Hybrid modeling of paper machine grade changes*. Ph. D. thesis, Helsinki University of Technology, Espoo, Finland.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems*. 2 ed. San Diego, USA: Academic Press.

AUTHOR BIOGRAPHIES

SEBASTIAN BOHLMANN is a Ph.D. candidate at Department of Simulation and Modelling - Institute of Systems Engineering at the Gottfried Wilhelm Leibniz Universität Hannover. He received a Dipl.-Ing. (FH) degree in mechatronics engineering from FHDW university of applied science. His research interests are machine learning and heuristic optimization algorithms, complex dynamic systems, control system synthesis and grid computing. His email address is <bohlmann@sim.uni-hannover.de>.

VOLKHARD KLINGER is a professor for embedded systems and computer science at the university of applied science FHDW in Hannover and Celle since 2002. After his academic studies at the RWTH Aachen he received his Ph.D. in Electrical Engineering from Technische Universität Hamburg-Harburg. During his 8-year research activity at the Technische Universität Hamburg-Harburg and research centres he focused on parallel and reconfigurable systems. Afterwards he developed ASICs and System-on-Chip solutions for cypher systems and fieldbus systems (x-by-wire). He teaches courses in computer science, embedded systems, electrical engineering and ASIC/system design. Beside his work in research and teachings, he works together with industrial partners in different applications with the main focus embedded systems. His email address is <Volkhard.Klinger@fhdw.de>.

HELENA SZCZERBICKA is head of the Department of Simulation and Modelling - Institute of Systems Engineering at the Gottfried Wilhelm Leibniz Universität Hannover. She received her Ph.D. in Engineering and her M.S in Applied Mathematics from the Warsaw University of Technology, Poland. Dr. Szczerbicka was formerly Professor for Computer Science at the University of Bremen, Germany. She teaches courses in discrete-event simulation, modelling methodology, queuing theory, stochastic Petri Nets, distributed simulation, computer organization and computer architecture. Her email address is <hsz@sim.uni-hannover.de>.