Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds

A NEW OBJECT-ORIENTED PETRI NET SIMULATION ENVIRONMENT BASED ON MODELICA

Sabrina Proß
Bernhard Bachmann

University of Applied Sciences Bielefeld
Am Stadtholz 24
D-33609 Bielefeld, GERMANY

Sebastian Jan Janowski
Ralf Hofestädt

Bielefeld University
Universitätsstr. 25
D-33615 Bielefeld, GERMANY

ABSTRACT

We present a new Petri net simulation environment to enable graphical hierarchical modeling, hybrid simulation, and animation of processes in life sciences, and technical applications, among others. In order to model these most different processes, a new powerful and universally usable mathematical modeling concept – extended Hybrid Petri (xHPN) – has been established. This specification is used for the Petri Net library (PNlib) realized by the object-oriented modeling language Modelica. In contrast to other approaches, we enable users to simultaneously reconstruct, analyze, and simulate complex dynamic models in one view. Therefore, we have connected the PNlib to VANESA, an open source tool for visualization and analysis of networks. Additionally, the PNlib is connected to Matlab/Simulink to use all the Matlab power for post-processing simulation results. To demonstrate this powerful environment, we are reporting our experience by modeling a technical and a general biological application case.

1 INTRODUCTION

Petri nets with their various extensions are a universal graphical modeling concept for representing processes from different application fields in nearly all degrees of abstraction. They support the qualitative modeling approach as well as the quantitative one. Once a qualitative Petri net model has been established, the quantitative data can be added successively. Furthermore, the processes can be modeled discretely as well as continuously and, in addition, discrete and continuous processes can also be combined within one Petri net model to so-called **hybrid Petri nets** (see e.g. (David and Alla 2001) and Figure 1). The Petri net formalism with all its extensions is so powerful that nearly all other formalisms are included. Hence, only one formalism is needed regardless of the chosen modeling approach (qualitative vs. quantitative, discrete vs. continuous, deterministic vs. stochastic), which is appropriate for the regarded system. Additionally, the Petri net formalism is easy to understand for researchers from different disciplines (biology, mathematics, informatics, engineers, business economists etc.) who work together in the modeling process. Thus, it is an ideal way for intuitive representing and communicating new knowledge of investigated systems. Besides, Petri nets allow hierarchical structuring of models and offer the possibility of different detailed views for every observer of the model.

In order to use Petri nets as a graphical modeling concept, Petri nets for their part have to be programmed by means of an appropriate language. The object-oriented modeling language **Modelica**, developed and promoted by the Modelica Association since 1996 for primarily modeling, simulation, and programming of physical and technical systems and processes (Modelica Association 2010), is ideally suited for this task. Modelica has become the de-facto standard for hybrid, multidisciplinary modeling. Each Petri net component, place and transition can be described object-oriented with the aid of a model in the Modelica language. These models are defined on the lowest level by discrete (event-based), differential,

978-1-4673-4781-5/12/$31.00 ©2012 IEEE

and algebraic equations (hybrid DAEs). An appropriate Modelica-tool enables graphical and hierarchical modeling, hybrid simulation, and animation.

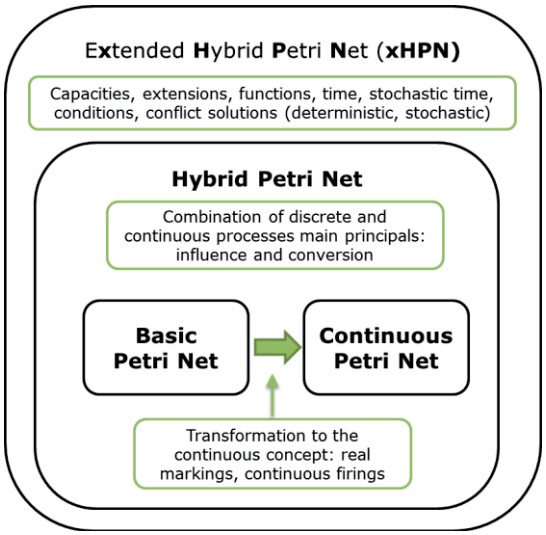

Figure 1: The relationships between the different Petri net formalisms.

To simplify the modeling process itself and, in addition, to give all involved researchers an adapted view of the model at a specific level of detail, models have to be constructed in a hierarchical structure. The Petri nets on the one hand and the Modelica language on the other enable hierarchical modeling concepts by **wrapping** the basic Petri net components to sub-models. These models can be used multiple times for specific processes in the same model, or across different models.

In addition to the modeling concept and the transformation to the Modelica language, appropriate mathematical methods are needed to **analyze** the established models. Thus, it is possible to determine model parameters, evaluate the robustness of the parameters towards small changes, simulate deterministically as well as stochastically, and optimize and control the underlying processes. Matlab/Simulink is ideally suited for performing these analysis. A bridge between Modelica models and Simulink is already available so that all the Matlab power can be used for the post-processing of the simulation results.

In order to show further powerful possibilities of the PNlib, we present the connection to **VANESA** (www.vanesa.sf.net), a software application for the modeling and simulation of biological processes. The software application makes use of the xHPN formalism realized in the PNlib to model and simulate biological systems based on project experimental data, information derived from integrated databases, and knowledge drawn by hand.

The paper is structured in the following way. At first related works are presented to clarify the demand of a new Petri net simulation environment. Afterwards, the developed Petri net formalism is introduced which has been used as specification for the implementation of the Petri nets components in the Modelica language. Implementation details, the connection to VANESA for modeling and visualization, and the connection to Matlab/Simulink for the post-processing of simulation results are part of section 4. Section 5 demonstrates two application cases based on the developed Petri net formalism. The first application demonstrates hierarchical modeling and hybrid simulation of a Senseo coffee machine within the PNlib. Further on, section 6 provides an example from the biological application field to present the connection to VANESA.

2 RELATED WORKS

Three different Petri net libraries are available on the Modelica homepage (www.modelica.org). The first was developed by Mosterman *et al.*, which enables the modeling of a restricted class of discrete Petri nets, called normal Petri nets (Mosterman et al. 1998). The places of normal Petri nets can only contain zero or

one token and all arcs can only be weighted with a value of one. External signals initiate the firing of transitions. Conflicts between several transitions are solved by priorities. The second Petri net library is an extension of the previous one and was developed by Fabricius (Fabricius 2001). The places can contain a non-negative integer number of tokens and be limited to non-negative integer minimum and maximum capacities. Furthermore, transitions are timed with fixed or stochastic delays. The third library, called StateGraph, is based on Grafcharts, which combine the function chart formalism of Grafcet with the hierarchical states of Statecharts (Johnsson and Årzén 1999). Transitions of state graphs have exactly one input and one output place. In addition, the places can only contain zero or one token and all arc weights are equal to one. The StateGraph library is part of the Modelica standard library and was developed by Otter et al. (Otter et al. 2005).

To enable modeling of most different kinds of processes with Petri nets in Modelica, the existing libraries have to be extended by following aspects:

- Transfer of the discrete Petri net concept to a continuous one,
- Support of edges with (functional) weightings,
- Support of test-, inhibitor, and read arcs,
- Support of different conflict resolutions (random decisions),
- Combination of discrete and continuous Petri net components to hybrid Petri nets.

Beside the Modelica language, three common tools are available for modeling processes with the hybrid Petri net formalism: Cell Illustrator, Snoopy, and SimHPN.

The **Cell Illustrator** is a commercial, widely-used tool available as a Java Web Start application that enables to draw, model, elucidate, and simulate complex biological processes and systems based on extended hybrid functional Petri nets (Nagasaki 2010). Discrete and continuous processes can be connected to perform hybrid simulations. The drawback of the Cell Illustrator is that the simulation is like a "black box". There is no information about how the Petri nets and the corresponding processes are defined, which are necessary for modeling and simulation, e.g. how conflicts in Petri nets are resolved, how the hybrid simulation is performed, and which integrators are used. In addition, there is no possibility to adapt solver settings in order to achieve reliable simulation results. The post-processing possibilities of simulation results for parameter estimation, process optimization etc. are also rather limited.

Snoopy is a freely available unifying Petri net framework to investigate biomolecular networks (Rohr 2010). A Petri net can be modeled time-free (qualitative model) or its behavior can be associated with time (quantitative model) such as stochastic, continuous, and hybrid Petri nets; thereby, different models are convertible into each other. It is also possible to structure the models hierarchically in order to manage complex networks. The drawback of Snoopy is that a continuous Petri net is interpreted as a graphical representation of a system of ordinary differential equations. Hence, the general Petri net property of non-negative marks cannot be held during simulation. Additionally, conflict situations of hybrid Petri nets are trapped not completely and, thus, negative markings can occur. Furthermore, places cannot be provided with capacities and no functions can be assigned to arcs in hybrid Petri nets.

SimHPN is a commercial MATLAB-based tool for modeling hybrid Petri nets (Júlvez and Mahulea 2012). The drawback of this tool is that not all conflicts occurring in hybrid Petri nets are solved and, additionally, test and inhibitory arc as well as capacities are not supported. Furthermore, there are no information available about processes important for simulation.

Hence, these problems led to the development of a new Petri net simulation environment specified by the established xHPNbio formalism. The xHPNbio elements are modeled object-oriented, which allows an easy way to maintain, extend, and modify them. Furthermore, the hybrid simulation is performed by an appropriate Modelica-tool. Using this approach it is even possible to adopt several solver settings to achieve more reliable simulation results. Moreover, the xHPNbio formalism is already integrated in VANESA, an easy-to-use biological modeling tool. Using VANESA scientists are able to reconstruct and simulate biological pathways either by drag-and-drop or by loading networks from databases. Based on the xHPNbio formalism reconstructed networks can be automatically translated into the Petri net language and later on simulated in one active window in VANESA. In Addition, a bridge from Modelica to

Matlab/Simulink is established to use all the Matlab-power for the post-processing of the simulation results.

3 EXTENDED HYBIRD PETRI NETS (XHPN)

The xHPN formalism comprises three different processes, called **transitions**: discrete, stochastic, and continuous transition, two different states, called **places**: discrete and continuous places, and four different arcs: normal, inhibitor, test, and read arc (see Figure 1). Discrete places contain a non-negative integer quantity, called **tokens** or **marks**, while continuous places contain a non-negative real quantity. These marks initiate transitions to fire according to specific conditions. These firings lead to changes of the marks in the connected places.

Discrete transitions are provided with **delays**. **Firing conditions** fire only then, when the associated delay is passed and the conditions are fulfilled. These fixed delays can be replaced by exponentially distributed random values. Such a corresponding transition is called **stochastic transition**. Thereby, the characteristic parameter λ of the exponential distribution can functionally depend on the markings of several places (cp. Heiner et al. 2008). It is recalculated at each point in time when the respective transition becomes active or when one or more markings of involved places change (cp. Proß et al. 2012). Based on the characteristic parameter, the next putative firing time $\tau = time + \mathrm{Exp}(\lambda)$ of the transition can be evaluated. Thus, it only fires when this point in time is reached.

Both – discrete and stochastic transitions - fire by removing the arc weight from all input places and adding the arc weight to all output places. In contrast, the firing of continuous transitions takes places as a continuous flow determined by the firing speed depended on markings and/or time.

Places and transitions are connected by "normal" arcs, which are weighted by non-negative integer and real numbers, respectively. But functions can also be written at the arcs depending on the current markings of the places and/or time. Places can also be connected to transitions by **test**, **inhibitor**, and **read** arcs. Then their markings do not change during the firing process. In the case of test and inhibitor arcs, the markings are only read to influence the time of firing, while read arcs only indicate the usage of the marking in the transition, e.g. for firing conditions or speed functions. If a place is connected to a transition by a test arc, the marking of the place must be greater than the arc weight to enable firing. If a place is connected to a transition by an inhibitor arc, the marking of the place must be less than the arc weight to enable firing. In both cases the markings of the places are not changed by firing. The same place can be connected to the same transition by a test and, in addition, by a normal arc as well as by an inhibitor and a normal arc. These arcs are called double arcs.

The **conversion** of a discrete to a continuous marking is realized by connecting a discrete transition to a continuous place and the conversion from a continuous to a discrete marking is realized by connecting a continuous place to a discrete transition. However, the conversion is always performed by discrete transitions. Discrete places can only influence the time when continuous transitions fire, whereas their marking cannot be changed during the firing process. It is important to mention that discrete transitions always fire in a discrete manner by removing and adding marks after a delay is passed, regardless of whether a discrete or a continuous place is connected to it. However, continuous transitions always fire in a continuous flow, so that a discrete place can only be connected to continuous transition if it is input as well as output of the transition with arcs of the same weight.

Summarized, an **xHPN** comprises of:

- discrete and continuous places,
- discrete, stochastic, and continuous transitions,
- places can be connected to transitions by normal, test, inhibitor, read arcs, and double arcs (test or inhibitor arc and normal arc), while transitions can only be connected to places by normal arcs,
- arc weights can be non-negative functions depending on markings and/or time,
- discrete places must be input and output of continuous transitions with arcs of the same weight,
- places can be provided with minimum and maximum capacities,
- discrete transitions can be provided with delays,

- stochastic transitions can be provided with hazard functions depending on the markings,
- continuous transitions can be provided with maximum speed functions depending on the markings and/or time,
- all transitions can be provided with additional firing conditions depending on all possible model variables, and
- conflicts of places are solved by providing the input and output transitions, respectively, with priorities or probabilities. These conflicts can occur between a discrete or continuous place and two or more discrete transitions, but also between a discrete place and two or more continuous transitions if the discrete place is input and output of the transitions with arcs of the same weights (cp. Proß and Bachmann 2012). If the discrete place is connected by test arcs instead of loop connections, there will be no conflict.

A formal definition of the xHPN-formalism and the corresponding processes is given in (Proß et al. 2012).

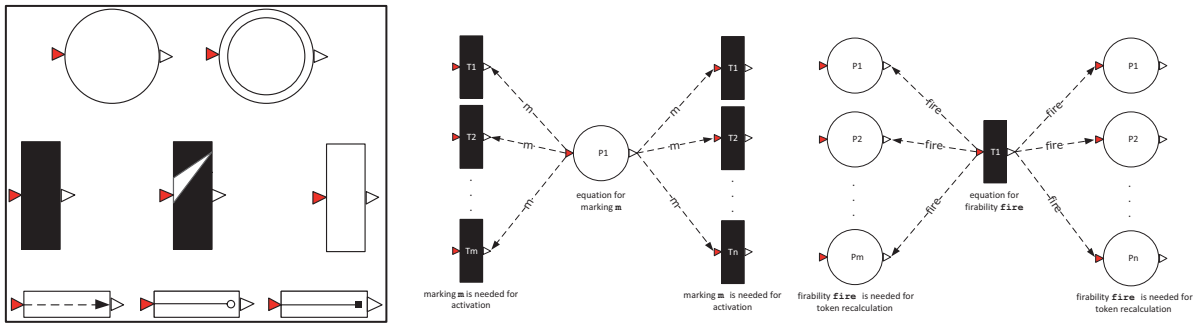

Figure 2: Left: Icons of the PNlib (from left to right); top: discrete and continuous place; middle: discrete, stochastic, and continuous transition; bottom: test, inhibitor, and read arc. Right: The connector variable `m` for the marking of a place is output of places and input of transitions (left). The connector variable `fire` for the firability of a transition is output of transitions and input of places (right).

4 PNLIB – A MODELICA BASED LIBRARY FOR MODELING XHPNS

The xHPN formalism and the corresponding definitions for activation, enabling, and firing (see Proß 2012), which are essential for simulation have been implemented by means of the object-oriented modeling language Modelica to enable graphical hierarchical modeling, hybrid simulation, and animation. Modelica is developed and promoted by the Modelica Association since 1996 for modeling, simulation, and programming of physical and technical systems and processes. Additionally, the Modelica standard library available from the Modelica Association is able to model mechanical (1D/3D), electrical (analog, digital, machines), thermal, fluid, control systems, and hierarchical state machines. Furthermore, several libraries have been developed in the last decade for specific applications. An overview can be found on the Modelica homepage (www.modelica.org). The development of the language and libraries is ongoing and driven by several European projects (EUROSYSLIB, MODELISAR, OPENPROD, and MODRIO). Since the year 2000, Modelica has been used successfully in industry, which is documented in the proceedings of many Modelica conferences and journals.

Modelica models are described on the textual level by discrete, differential, and algebraic equations (**hybrid DAEs**) and by schematics on the graphical level. A schematic consists of connected components which are defined by other components. On the lowest level they are connected by equations in the Modelica syntax. Therefore, the components have connectors, which describe the interaction between them. By drawing a line from one component to another, a connection is established to enable interactions. In this manner a model is constructed. Several components can be structured in libraries, called packages, which provide hierarchical modeling.

Moreover, the **wrapping technique** enables the representation of sub-models consisting of several connected components by a specific adapted icon in order to simplify the modeling process. These sub-models can be used multiple times in the same model or across different models. In addition, this approach offers an easy-to-use-model at the top level with an intuitive and familiar adapted view.

However, for graphical modeling, simulation, and animation an appropriated environment is needed. Several commercial and open- source tools are available. A full list can be found on the Modelica homepage (www.modelica.org).

4.1 Implementation

Each of the xHPN components - transitions, places, and arcs - is modeled by its own Modelica model, which is organized and structured in a Modelica package, called **PNlib** (**P**etri **N**et **lib**rary). All components are defined on the lowest level by discrete (event-based), algebraic, and differential equations (cp. Proß and Bachmann 2012). The object-oriented modeling of places, transitions, and arcs can only be realized if the Petri net components are able to interchange variables. Modelica makes this interaction possible by means of the specialized class `connector`. A connector comprises variables, which are calculated in one component, but also needed in the connected components for further calculations. The connectors of the Petri net component models are represented by red and white triangles with Petri net icons (see Figure 2 left). Figure 2 (right) shows two examples of connector variables. The current marking of a place `m` is calculated in the place model. However, it is also needed in the connected input and output transitions to determine if they can become active. Hence, it is an output of the place connectors and an input of the transition connectors. On the other hand, the variable `fire` is determined in the transitions, but also needed in the connected places for the recalculation of the marking. Hence, it is an output of the transition connectors and an input of the place connectors.

The main process of the place model is the recalculation of the marking after firing a connected transition. In the case of the discrete place model, this is realized by the discrete equation

```
when fire or reStart then
   m = if fire then pre(m) + firingSumIn - firingSumOut else reStartTokens;
end when;
```

whereby `pre(m)` accesses the marking `t` immediately before the transitions fire. To this amount, the arc weight sum of all firing input transitions is added and the arc weight sum of all firing output transitions is subtracted from it. Additionally, the marking is reset to `reStartTokens` when the user-defined Boolean condition `reStart` becomes true. The marking of continuous places can change continuously as well as discretely. This is implemented by the following construct:

```
der(m) = conMarkChange;
when discreteFire then
   reinit(m, m+discreteMarkChange);
end when;
when reStart then
   reinit(m, reStartMarks);
end when;
```

whereby the `der`-operator accesses the derivative of the marking `m` according to time. The continuous mark change is performed by a differential equation, while the discrete mark change is performed by the `reinit`-operator within a discrete equation. This operator causes a re-initialization of the continuous marking each time a connected discrete transition fires. Additionally, the marking is re-initialized by `reStartMarks` when the user-defined Boolean condition `reStart` becomes true. Via the connector variable `m`, places report their current markings to the transitions (see Figure 2 right). Based on the current markings of the places, it is checked in the transition model by an algorithmic procedure if the transition can become active. A discrete transition waits until the delay is passed and a stochastic transition waits till the next putative firing time is reached before firing. Based on this information, the places enable some of the active transitions to fire. At this point, several conflicts can occur which have to be resolved appropriately by the methods mentioned in (Proß 2012, Proß and Bachmann 2012) to get a successful and

reliable simulation. When a transition is enabled by all its connected places, it is firable and reports this via the connector variable `fire` to the connected places (see Figure 2 right). Then, the places recalculate their markings based on this new information.

4.2 Connection to VANESA for modeling and visualization

VANESA is a biology-oriented software application with which biological scientists can intuitively model and simulate complex dynamic interactions and processes (www.vanesa.sf.net). Therefore, it combines different fields of studies, such as life science, database consulting, modeling and visualization for a semi-automatic and lab-validated reconstruction of biological networks. Using VANESA scientists are provided with theoretical supports, technologies, and tools to reconstruct and simulate biological phenomena *in silico*. Primarily, it is intended for biological scientists working at the bench.

For the modeling and simulation of biological processes, VANESA makes use of the xHPN formalism realized in the Modelica library PNlib. Therefore, it provides a biologically sophisticated GUI in which biological models can be modeled and simulated based on project experimental data, information derived from integrated databases, or knowledge drawn by hand. Thus, any kind of biological model and network can be reconstructed and automatically converted into the language of Petri nets within VANESA in order to check behavioral and structural properties. Furthermore, it is possible to move between the different classes of modeling and simulation concepts, since the graphical user interface adapts automatically to the net class in the active window.

Sophisticated simulations can be performed using qualitative, stochastic, continuous, hybrid and functional modeling features of the xHPN formalism. Simulation results are available as tables and also visualized in diagrams within the graphical user interface of VANESA, showing the evolution over time of the token numbers on selected places. Thus, VANESA and the PNlib provide a well-defined ground to investigate dynamic models in various complementary ways. The animation can be triggered manually or within the active window.

Due to VANESAs generic design and strict separation of internal data structure and graphical representation, it is possible to easily convert the integrated ontology for network representation into the xHPN formalism that can be further simulated within the PNlib. Furthermore, it is easy to extend VANESA and the PNlib by new graph classes applying reuse and specialization of existing elements.

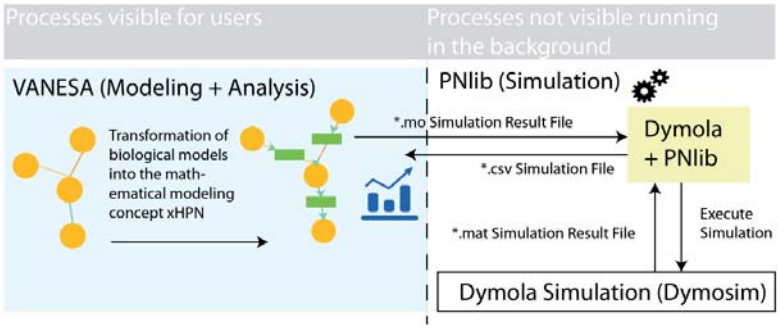

Figure 3: A pictogram of the simulation/communication bridge between VANESA and the PNlib in Modelica (Dymola)

In general, the communication between VANESA and the PNlib in Modelica is realized by a digital communication bridge running in the background as depicted in Figure 3. In order to perform a simulation users only need to provide their reconstructed biological systems with Petri net parameters, such as initial markings, and arc weights. Using the simulation functionality of VANESA, a script automatically translates the ready Petri net model into the appropriate .mo data exchange format and starts Modelica (Dymola) and the corresponding PNlib in the background for simulation processing. As soon as simulation results are available, simulation outcomes are automatically loaded in VANESA. The data and simu-

lation exchange is realized by a .csv data exchange, which lists progress steps and information in detail. Finally, results are matched on the network and made visible to users.

4.3 Connection to Matlab/Simulink for post-processing simulation results

Model construction based on such strong formalism as the xHPN formalism alone is not enough for a good working model. Usually, the constructed model comprises several parameters, which have to be estimated. This is particularly the case in the biological application field. Several biological databases are available, which summarize specific parameters, such as the biological database BRENDA that provides a collection of enzyme functional data. However, if the required parameters are not listed in databases or publications, they have to be estimated by experiments. But sometimes these experiments are too expensive, too imprecise, or not even feasible. In these cases, specific mathematical optimization methods can provide a means to adapt the model behavior as well as possible for the given experimental data. This procedure is called **parameter estimation**. Parameter estimation engenders an optimization problem: *Minimize an objective function which represents the goodness of a parameter set*. This objective function can be formulated mathematically by a non-linear programming problem constrained by the hybrid DAEs of the Modelica model and upper and lower bounds for every parameter. These objective functions in combination with an xHPN model are not only non-linear but also usually discontinuous and not-differentiable due to the discrete changes of hybrid Petri nets. Due to the non-differentiability, the usage of methods which determine decent directions from derivatives of the objective function is not possible.

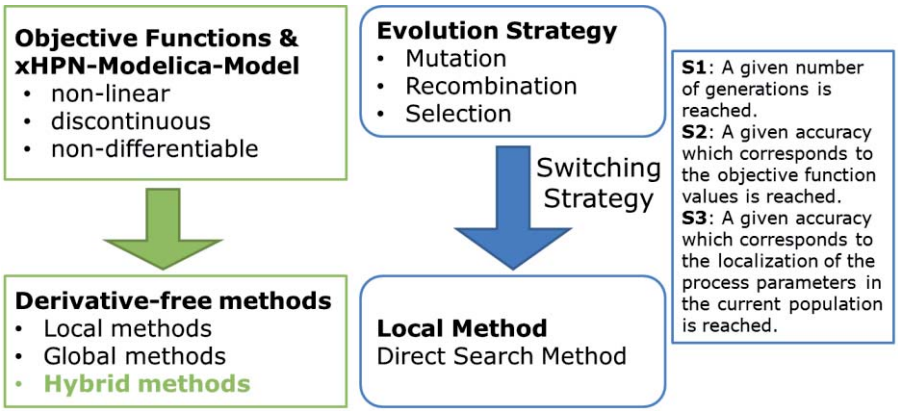

Figure 4: Parameter estimation of Modelica models

However, derivative-free methods do not require derivatives to minimize the objective function and, hence, these methods are applicable. It has to be distinguished between local and global methods. **Local methods** try to find the minimum starting from a given point. Thereby, only local information about the objective function from the neighborhood of the current approximation is used to update the approximation; hence, the global structure of an objective function is unknown to a local method. Additionally, it is usually expected that such methods converge to the local minimum, which is nearest to the starting point. However, the objective of **global methods** is to find the global minimum of the optimization problem usually in the presence of multiple local minima by seeking the whole search space. Global methods can be further divided into deterministic and stochastic methods. **Deterministic methods** always achieve the same result starting from the same setting while **stochastic methods** involve random mechanisms. Local and global methods can also be combined to **hybrid methods,** which should avoid the high computational costs of global methods due to their slow convergence near the minimum. Additionally, the entrapment in a local minimum should be prevented, which is often the drawback of local methods. The best results have been found by combing the global **Evolution Strategy** with the local Hooke-Jeeves method. After one of a set of specific switching criteria are fulfilled, the algorithm switches from the global search to the local one (cp. Proß 2012).

Directly linked to parameter estimation is the **sensitivity analysis** of model parameters that aims to identify, optimize, reduce, and verify the model. Furthermore, particular mathematical optimization methods enable the optimization of biological processes, called **process optimization**. This plays an important role in industrial biotechnology. Based on a model, it is possible to control biological processes in the best possible way and to gain such maximum product yields from the cultivated organisms. To realize the analysis of Modelica models by mathematical methods, they have to be simulated several times with different parameter settings. Then, the arising simulation results are post-processed by algorithmic procedures in Matlab. Therefore, the Modelica models in Dymola are connected to Matlab/Simulink, which is accomplished with the aid of a Dymola interface in Simulink and a set of Matlab m-files utilities (see Dynasim AB 2010 for a detailed description). An established Matlab-tool, called **AMMod** (**A**nalysis of **M**odelica **Mod**els), already provides several mathematical methods for data preprocessing, relationship analysis, parameter estimation, sensitivity analysis, deterministic and stochastic hybrid simulation, and process optimization (cp. Proß 2012).

5 APPLICATION CASE 1 – MODELING A SENSEO COFFEE MACHINE

The main feature of a Senseo coffee machine is that the coffee is placed in the machine in a pre-portioned form by so-called coffee pads. One pad is generally used to make one cup of coffee (125°ml) and two pads sufficient for two 125 ml cups or one large 250 ml cup. After a warm-up time of about 60 seconds and the insertion of a coffee pad, the coffee can be made. In this warm-up phase, the water is heated to 90°C and then pressed with a pressure of about 1.4 bar within 40 seconds through the pad. In contrast to a normal coffee machine that boils the water continuously and transports it by its own buoyancy (hot bubbles) up into the filter, the Senseo machine heats a portion of water completely in a heating chamber and then pumps it through the pad. To ensure that the heating chamber in the machine is always filled with water, a float is placed in the removable water tank, which allows measuring the minimal capacity. If the minimum level is exceeded, the heater is turned off. If there is a sufficient water level, the next portion of water is heated directly after the boiling and filling. Summarized, a part of the aforementioned described functional principles and the corresponding xHPN definitions are listed in Table 1.

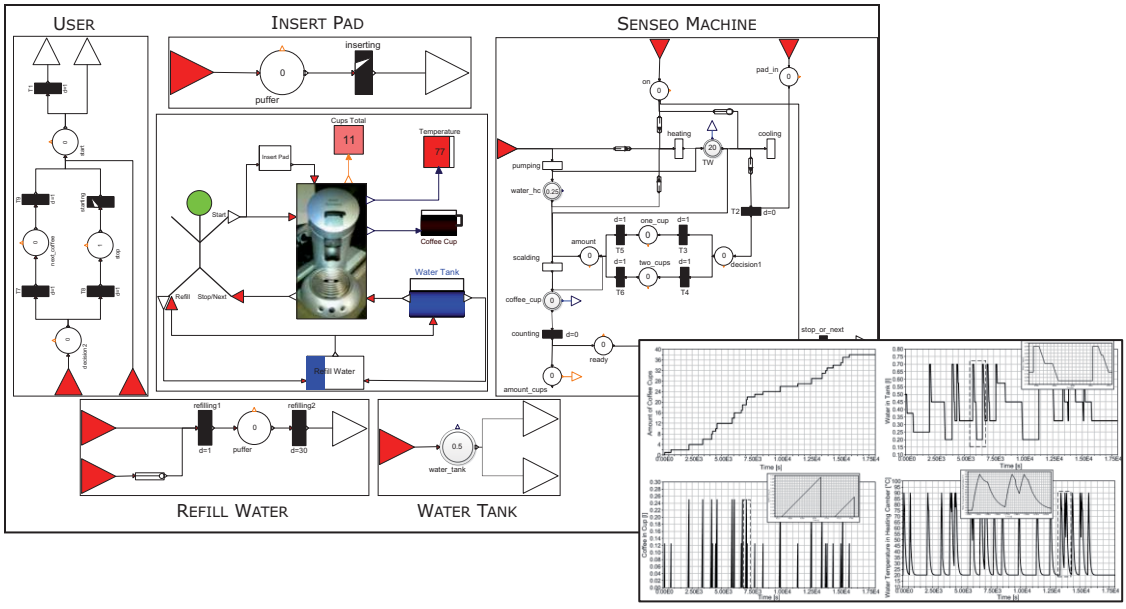

Figure 5: Wrapping technique: sub-models of the Senseo coffee machine model and simulation results

The xHPN model of the Senseo coffee machine has been implemented in a hierarchical structure with the aid of the PNlib and the Modelica tool Dymola.

Figure 5 (middle) shows the model at the top level. An animation of this model displays the amount of produced coffees as well as the current filling level of the coffee cup and the water tank. Additionally, the head of the stickman is green when the machine is on and red when it is off. When a pad is inserted the respective rectangle is brown and, similarly, when water is refilled the respective rectangle is blue.

Table 1: Part of the representation of a Senseo coffee machine by an xHPN model (first 4 steps)

Step	Functional Principle	xHPN model
1	The machine is off.	One token is placed in place *start.*
2	The time between turning the machine on and off again takes an average of 1200s.	The stochastic transition *starting* has a hazard function of 1/1200.
3	If the machine is started and the water in the tank is less than 0.35 l, the tank has to be refilled. The tank is always filled completely and this procedure takes 30s. The water tank has a maximum capacity of 0.7 l.	The place *water_tank* is connected to the transition *refilling1* by an inhibitor arc with a weight of 0.35. The transition *refilling2* has a delay of 30 and the weight of the arc from this transition back to the place *water_tank* is (*0.7-water_tank*), where *water_tank* is the current marking of place *water_tank*. The place *water_tank* has a maximum capacity of 0.7.
4	If there is enough water in the tank, the water in the heating camber is heated. The heating speed (h_s) depends on the power of the Senseo machine (P_{senseo}), the specific heat capacity of water (c), and the amount of water to be heated in the heating camber (W_{HC}): $$h_s = \frac{P_{senseo}}{c \cdot W_{HC}}.$$ The maximum temperature of the water is 90°C.	The place *water_tank* is connected to the transition *heating* by a test arc with the weight 0.35. The continuous transitions *heating* has a maximum speed of *Psenso/(c*waterHC)*, where *waterHC* is the current marking of the place *waterHC*. The maximum capacity of place *TW* is set to 90.

6 APPLICATION CASE 2 – MODELING GENETIC PATHWAYS

In order to demonstrate the functionality and features of the PNlib in VANESA, a genetic regulatory mechanism example is presented that shows how a basic transcription-regulation in prokaryotes can be modeled and simulated. In general, gene expression in prokaryotes is often responsive to signal molecules of the nutritional or environmental conditions affecting the cell.

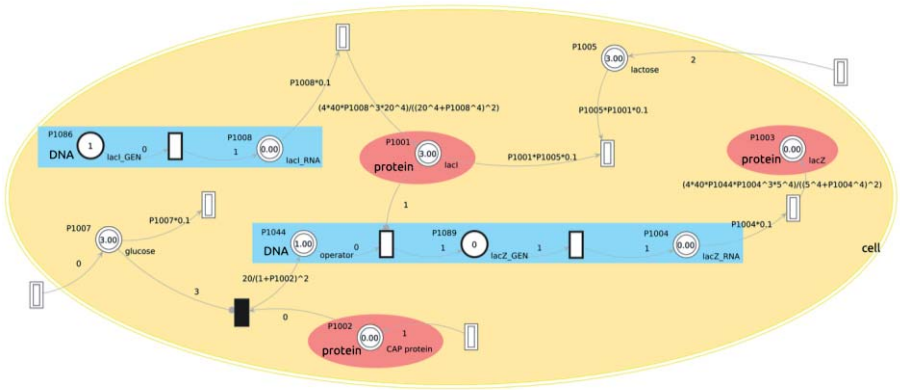

Figure 6: Petri Net model for the simulation of the transcription-regulated lac-operon system of the bacterium *Escherichia coli.*

Interested in these processes, we present one part of the signaling network of the transcription-regulated lac-operon system of the bacterium *Escherichia coli*. The focus of this modeling and simulation example is the gene induction. Processes termed induction describe increased synthesis of enzymes in response to the presence of a particular substrate. Therefore, the presented example (see Figure 6) simulates the cell behavior of the bacterium *Escherichia coli* in response to decreasing glucose and increasing lactose in the cell environment.

The *lac*-operon is responsible for the transport and metabolism of lactose in *Escherichia coli*. The bacterium can use lactose as both a carbon and energy source. If glucose is absent and lactose available the synthesis of β-galactosidase is induced by activating the lac-operon. If two energy sources are available such as glucose and lactose, the more readily-available energy source, in this case glucose, is used.

The *lac*-operon consists of three adjacent structural genes, *lacZ*, *lacY*, and *lacA*. Of particular interest is the *lacZ* gene, a structural gene for β-galactosidase, which is fully synthesized when lactose is present. If lactose is not available, the *lacI* gene inhibits the operon by blocking the promoter. Thus, the RNA-Polymerase is not able to bind to the promoter. Furthermore, if both glucose and lactose are available the CAP protein (catabolite activator protein) is inhibited, which increases the binding affinity of the RNA-Polymerase to the lac-operon.

The simulation results presented in Figure 7, clearly show how the *lacZ* transcription increases with absence of glucose and availability of lactose. Therefore, the CAP protein is no longer inhibited and the *lacI* gene effect decreases. The promoter becomes active and lactose consumption begins.

Petri Net parameters, such as initial markings, and arc weights are based on Hill functions for activator and repressors, and typical parameter values for *Escherichia coli*, such as transitions between protein states, timescales for the equilibrium binding of small molecules to proteins, and timescales of transcription factor binding to DNA, among others.

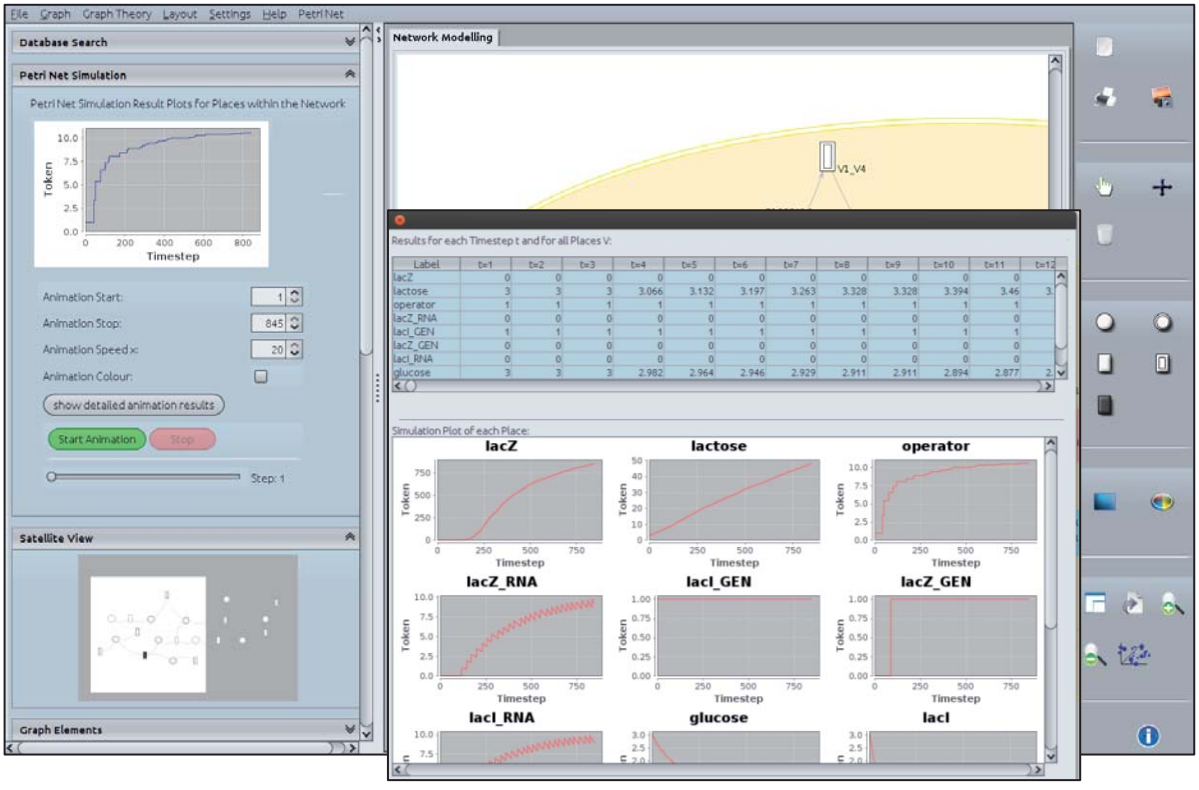

Figure 7: Simulation results of the transcription-regulated lac-operon system of the bacterium *Escherichia coli* within VANESA.

7 CONCLUSIONS

A powerful Petri net environment has been developed for graphical hierarchical modeling and hybrid simulation as well as animation of processes from most different application fields. Thereby, the mathematical modeling concept xHPN serves as specification for performing hybrid simulations. The xHPN elements are modeled object-oriented by discrete, differential, and algebraic equations in the Modelica language. This is an easy way to maintain, extend, and modify the components. The hybrid simulation is performed with an appropriate Modelica-tool, which has several possibilities to adapt solver settings in order to achieve reliable simulation results.

The connection to VANESA enables the modeling, editing, visualization, and animation of xHPN-models with an easy-to-use interface. Thus, users are able to intuitively model and simulate complex dynamic interactions and processes within a biology-oriented software application. Therefore, users are provided with different fields of studies, such as life science, database consulting, modeling and visualization for a semi-automatic and lab-validated reconstruction of biological networks.

Moreover, the connection to Matlab/Simulink offers the whole Matlab power for the post-processing of the simulation results of Modelica models. The Matlab-based tool AMMod (Analysis of Modelica Models) provides several mathematical methods for data preprocessing, relationship analysis, parameter estimation, sensitivity analysis, deterministic and stochastic hybrid simulation, and process optimization.

The application of the new Petri net simulation environment was demonstrated with a model of a Senseo coffee machine in order to show the applicability of the xHPN formalism as well as the graphical hierarchical modeling and hybrid simulation with the PNlib. Moreover, the example of one genetic regulatory mechanism from the biological application field presented the connection to VANESA and possibilities to model and simulate biological networks in one software application.

A future goal is to provide an open source Petri-net simulation tool. This demands further development of the open source Modelica-tool OpenModelica to get the PNlib to work with it as some Modelica features are not supported. Moreover, the xHPN formalism as well as the PNlib will be extended by fuzzy logic and the color concept to further enhance the range of application fields.

REFERENCES

David R, Alla H. 2001. On Hybrid Petri Nets. Discrete Event Dynamic Systems: *Theory and Applications*(11): 9–40.

Dynasim AB. 2010. Dymola-Dynamic Modeling Laboratory-User Manual Volume 2, Lund, Sweden

Fabricius S.M. 2001. Extensions to the Petri Net Library in Modelica. ETH Zurich, Switzerland.

Heiner M., Gilbert D., Donaldson R. 2008. Petri nets for systems and synthetic biology. In *Proceedings 8th International Conference on Formal Methods for Computational Systems Biology*:215–264.

Johnsson C., Årzén, Grafchart and grafcet. 1999. A comparison between two graphical languages aimed for sequential control applications, *Preprints 14th World Congress of IFAC(A)*: 19-24.

Júlvez J., Mahulea C. 2012. SimHPN: A MATLAB toolbox for simulation, analysis and design with hybrid Petri nets. *Nonlinear Analysis: Hybrid Systems* 6(2):806-817.

Modelica Association. 2010. Modelica - A Unified Object-Oriented Language for Physical Systems Modeling Language Specification Version 3.2.

Mosterman PJ, Otter M, Elmqvist H. 1998. Modeling Petri nets as local constraint equations for hybrid systems using Modelica. In *Proceedings of SCS Summer Simulation Conference*:314–319.

Nagasaki M., Saito A., Jeong E., Li C., Kojima K., Ikeda E., Miyano S. 2010. Cell Illustrator 4.0: A computational platform for systems biology. *In Silico Biology* 10(1), 5–26.

Otter M, Årzén KE, Dressler I. 2005. StateGraph-a Modelica library for hierarchical state machines. In *Proceedings of 4th International Modelica Conference*:21-33.

Proß S. 2012. Hybrid Modeling and Optimization of Biological Processes. *PhD thesis* (in preparation). Faculty of Technology, Bielefeld University, Germany.

Proß S., Bachmann B. 2012. PNlib – An Advanced Petri Library for Hybrid Process Modeling, In *Proceedings of the Modelica Conference* (accepted), Munich, Germany.

Proß S., Janowski S. J., Bachmann B., Kaltschmidt C., Kaltschmidt B. PNlib - A Modelica Library for Simulation of Biological Systems based on Extended Hybrid Petri Nets, 3rd International Workshop on Biological Processes & Petri Nets, Hamburg, Germany, 2012.

Rohr C., Marwan W., Heiner M. 2012. Snoopy—a unifying Petri net framework to investigate biomolecular networks. *Bioinformatics* 26(7), 974.

AUTHOR BIOGRAPHIES

SABRINA PROß is a final-year PhD student at the University of Applied Sciences, FH Bielefeld Germany, who studied mathematics. In 2008 she has started her thesis on the hybrid modeling of biological systems. Therefore, she established, developed and used new Petri net paradigms and libraries for the simulation and parameter estimation of biological models. Her email address is sabrina.pross@fh-bielefeld.de.

SEBASTIAN JAN JANOWSKI is final-year PhD student at the Faculty of Technology, Bielefeld University, Germany, who studied Bioinformatics and Genome research at the Department for Bioinformatics and Medical Informatics. In 2010 he has started his thesis on the modeling and visualization of biomedical networks. He is mainly interested in the database-driven network reconstruction and modeling. His email address is janowski@uni-bielefeld.de.

BERNHARD BACHMANN is a professor for Engineering and Mathematics at the University of Applied Sciences, FH Bielefeld, Germany. His research interest is the applied mathematical modeling and optimization. He is a member of the Modelica Association, which promotes and develops a non-proprietary, object-oriented, equation based language to conveniently model complex physical systems. His email address is bernhard.bachmann@fh-bielefeld.de.

RALF HOFESTÄDT is a professor for Bioinformatics and Medical Informatics at the Faculty of Technology, Bielefeld University, Germany. His research interest is the modeling and simulation of metabolic networks, integration of molecular data, the application of molecular data warehouses, medical diagnosis systems, and detection of metabolic diseases. His email address is ralf.hofestaedt@uni-bielefeld.de.