Modeling and Evaluation of Stochastic Petri Nets With TimeNET 4.1

Armin Zimmermann
System and Software Engineering Group
Ilmenau University of Technology
Ilmenau, Germany
Email: armin.zimmermann@tu-ilmenau.de

***Abstract*—The paper presents a major update of the software tool TimeNET, a package for the modeling and performance evaluation of standard and colored stochastic Petri nets. Among its main characteristics are simulation and analysis modules for stationary and transient evaluation of Petri nets including non-exponentially distributed delays, as well as an efficient simulation module for complex colored models.**

An overview of the tool is given as well as a description of the new features, which are demonstrated using a manufacturing system application example. The tool is available free of charge for non-commercial use.

Keywords: Modeling tool, TimeNET, stochastic Petri nets, colored Petri nets, performance evaluation

I. Introduction

The design of non-trivial technical systems is more efficient and less risky when the effect of design decisions can be estimated early. Model-based performance evaluation is an important help. In the area of computer-controlled technical systems, numerous model classes within the field of *stochastic discrete event systems* have been proposed. Stochastic Petri nets and their variants are widely accepted today as a mature tool for describing systems exhibiting concurrency, resource restrictions, synchronization, time-dependency, and probabilistic behavior. However, their usability in practice depends on the availability of powerful software tools that are easy to use. A graphical user interface is required to efficiently enter the graphical representation. The task of improving a model until it behaves as expected can be supported by qualitative checks on the model as well as an interactive simulation or visualization of the behavior. Finally, evaluation algorithms and other aspects such as code generation or direct control have to be implemented.

TimeNET is a software tool for the modeling and performability evaluation using stochastic Petri nets. Its two main characteristics are the evaluation of models with non-exponentially distributed firing delays and the ability to model and evaluate complex colored stochastic Petri nets. The latter model type combines colored Petri nets with freely definable object-like tokens with the probability and time concept that is generally accepted for stochastic Petri nets such as GSPNs (generalized stochastic Petri nets [1]). The tool is thus capable of modeling complex systems in a compact way, and to obtrain performance measures for them. TimeNET has been successfully applied during several modeling and performance evaluation projects and there are several hundred installations in universities and other organizations worldwide.

The tool is available free of charge for non-commercial use. To get a copy of the tool, follow the instructions on the web page `http://www.tu-ilmenau.de/sse/timenet/`. There are regular updates with minor additions and bug releases made available at the tool home page. A comprehensive user manual dan be downloaded as well.

This paper describes version 4.1 of the TimeNET tool, which is available since April 2012. A brief history is given in the sequel, followed by an overview of related software packages. Section II describes the main features of TimeNET. Specifically, the supported net classes and corresponding analysis modules are mentioned; a description of user interface and software architecture is given, and the section ends with a list of applications with references.

Section III describes the new features of TimeNET 4.1. Among the most important improvements are stationary and transient simulation modules with statistical accuracy control, extended color-dependencies for inscriptions, and rare-event simulation (all for stochastic colored Petri net models); and non-interactive token game visualization, better graphical display of analysis results, and PNML exchange format [2] import and export (all for eDSPN models). The tool in general is now available as a self-contained installation program, has been ported to Windows 7, and improved by a help system inside the user interface.

An example demonstrates some of the improvements in Section IV. A manufacturing system model from the literature is used, which shows how compact the behavior can be specified with a colored model in comparison to the original generalized stochastic Petri net. The paper ends with conclusions and acknowledgements.

History of TimeNET: The development of TimeNET started in the mid-1990s, based on an earlier implementation of DSPNexpress [3], both at Technische Universität Berlin. Modeling and evaluation of extended deterministic and stochastic Petri nets (eDSPNs) were added. The graphical user interface was originally based on the X Athena Widget toolkit, and not easily adaptable to extensions. TimeNET 3 [4] contained a major change with a new Motif-based user interface Motif and ability to capture several modeling environments within the same (generic) tool. This allowed for extensions included

VALUETOOLS 2012, October 09-12, Cargèse, France

DOI 10.4108/valuetools.2012.250263

variable-free colored Petri nets, fluid stochastic Petri nets, discrete-time stochastic Petri nets, and modular blocks of SPNs. Other extensions dealt with specialized analysis algorithms for existing model classes.

The next major development step was TimeNET 4 [5], [6] in 2007, which included modeling and simulation capabilities for colored stochastic Petri nets. A new graphical user interface was implemented in Java, making the whole tool available both in Linux and Windows environments. Model type descriptions and models themselves are stored in an application-specific XML format since then. Some prototypical modules of the tool are discontinued with TimeNET 4 to reduce maintenance efforts, including discrete-time models, fluid models, and variable-free colored nets (which are replaced by the broader class of SCPN now).

Tool development has moved to Technische Universität Ilmenau in 2008, and version 4.1 has been released in April 2012. This paper covers the improvements contained in this update (c.f. Section III).

Related Software Tools: Numerous commercial, research and prototype software tools have been developed for stochastic Petri nets and related formalisms in the past; overviews are given in [7], [8]. Probably the best starting point for a search is the Petri net home page [9]. A non-exhaustive overview of related tools is given here.

GSPN models are graphically edited, analyzed or simulated with the software package GreatSPN 2.0 [10]. The tool provides a graphical user interface for editing and evaluating generalized stochastic Petri net models. Moreover, stochastic well-formed nets (SWN) with colored tokens can be evaluated. Symmetries in the reachability graph can be detected and lumping techniques facilitate analysis algorithms with less computational effort in many cases. Components for direct numerical analysis and simulation (transient and steady-state) are contained.

Some analysis methods and a graphical user interface for GSPN models have been implemented in the PIPE tool [11].

The software package SPNP 6 [12] has been developed at Duke University. It contains components for the transient and steady-state analysis of stochastic reward nets (SRNs), which are comparable to GSPNs. Models are specified alphanumerically as C functions. Very general performance measure specifications can be given and a sensitivity analysis module is available. Transient and cumulative reward measures are obtained using randomization. The tool has been extended with a Tcl/Tk based graphical user interface, which is referred to as iSPN.

The tool WebSPN 3.3 [13] analyzes non-Markovian stochastic Petri nets, and has a web-enabled user interface. Phase-type fitted functions may be used to specify non-Markovian delays. The tool can map UML models to Petri nets.

Advanced methods for the analysis of non-Markovian Petri net models, especially symbolic state-space analysis of preemptive Time Petri nets, have been implemented in the Oris tool [14].

Colored Petri nets [15] with ML-based inscriptions are supported by the CPN-Tools [16]. The tool is not primarily aiming at performance evaluation, but offers a time notion embedded in tokens for simulation which is different from the usual understanding of time in SPN.

Other approaches to tool implementations have been towards the integration of other tools in one common user interface, and by combining the results of the individual tools.

One example is the SHARPE tool (Symbolic Hierarchical Automated Reliability and Performance Evaluator [17]). It can be used for specifying and analyzing performance, reliability and performability models. The toolkit provides a graphical user interface, specification language, and solution methods e.g. for fault-trees, queuing networks, semi-Markov reward models, and GSPNs. Steady-state, transient and interval measures can be computed and used in other models.

OpenSESAME [18] covers reliability block diagrams with inter-block dependencies to evaluate the reliability of complex fault-tolerant systems.

Different net classes, including colored models, can be edited with the Snoopy tool [19]. It can be used to simulate models and uses analysis modules of several other tools to gain results. Its main application area is Petri net models of biochemical processes.

The software package HiQPN [20] has been developed at the University of Dortmund. It uses the model class queuing Petri nets (QPNs) as well as their hierarchical combination (HiQPNs), including colored Petri net tokens and queuing places. Several analysis techniques for the steady-state performance evaluation are provided, namely decomposition approaches based on tensor algebra. The hierarchical structure of the model is used for a structured description of the stochastic generator matrix. QPME [21] is a recent Java-based simulator and editor for this net class.

Möbius 2.3 [22] is a multi-formalism, multi-solution software tool, in which model classes as well as analysis algorithms can be combined. This is done using an abstract model description. Its implementation is consequently done as an abstract functional interface. The Möbius framework is able to integrate additional solvers that are applicable to some of its model classes in a modular way, and thus relieves the developers of prototype implementations from designing a complete tool from scratch. On the other hand it is possible to use existing evaluation algorithms once a new model class has been described with the abstract functional interface. A significant point of Möbius is the possibility to combine model parts from different classes, and a simultaneous simulator for such models.

II. Overview of TimeNET

This section gives a brief overview of TimeNET's model classes and available analysis methods (Subsection II-A). Moreover, its graphical user interface, the underlying software architecture, and some application areas are presented in Subsections II-B, II-C, and II-D.

A. Net Classes and Analysis Methods

TimeNET supports modeling and evaluation of different net classes, namely

- eDSPNs — extended deterministic and stochastic Petri nets, which include generalized stochastic Petri nets as a special case;
- SCPNs — colored stochastic Petri nets;
- and stochastic UML state charts (prototype status).

Characteristics of the model classes and analysis modules available in the tool are listed below.

The classic main model class of TimeNET are eDSPNs. In this net class with indistinguishable tokens, firing delays of transitions can either be zero (immediate), exponentially distributed, deterministic, or belong to a class of general distributions called *expolynomial*. Such a distribution function can be piecewise defined by exponential polynomials and has finite support. It may also contain jumps, making it possible to mix discrete and continuous components. Many known distributions (uniform, triangular, truncated exponential, finite discrete) belong to this class. A textual syntax is used to define such delay functions.

If all transitions with non-exponentially distributed firing times are mutually exclusive in a model, stationary numerical analysis is possible [23]. If the non-exponentially timed transitions are restricted to have deterministic firing times, transient numerical analysis is also provided [24], [25]. If there are only immediate and exponentially timed transitions, the model is a GSPN and standard algorithms for steady-state and transient numerical evaluation based on an isomorphic Markov chain are applicable.

Structural properties of eDSPNs like extended conflict sets and invariants (semi flows) can be obtained with TimeNET. They are displayed and checked by the modeler to examine the correct model specification [26]. For a steady-state or transient analysis of a simple stochastic Petri net model of any kind, the reachability graph is computed. Structural properties are exploited for an efficient generation of the reachability graph. Subnets of immediate transitions are evaluated in isolation.

Transient analysis of DSPNs is based on supplementary variables [24], [25], which capture the elapsed enabling time of transitions with non-exponentially distributed firing delays. TimeNET shows the evolution of the performance measures from the initial marking up to the transient time graphically during a transient analysis.

The tool also comprises a simulation component for eDSPN models [27], which is not subject to the restriction of only one enabled non-Markovian transition per marking. Steady-state and transient simulation algorithms are available. Results can be obtained faster by parallel replications (master / slave architecture), using control variates. During the simulation run, intermediate results of the performance measures are displayed graphically together with the confidence intervals.

In safety-critical systems, catastrophic events are important, but happen only rarely. Their standard simulation up to a certain statistical confidence is thus unacceptably time-consuming. The field of rare-event simulation covers this issue, and a technically attractive solution is the RESTART algorithm [28]. The RESTART implementation in TimeNET was done for restricted problem types of simple stochastic Petri nets first [29]. Some of the restrictions have been overcome by allowing more general performance measures [30] later.

An automatic optimization method for eDSPN models based on a two-phase simulated annealing has been implemented as a prototypical extension of TimeNET [31], which is not part of the distributed version.

A TimeNET model class capturing UML Statecharts has been added recently [32]. Such a model is translated into an eDSPN for performance evaluation.

Stochastic colored Petri nets (SCPNs) are available since TimeNET 4 [33]. Due to the inherent complexity of the models, a requirement of only one non-exponential transition per marking was decided to be too restrictive. Thus only simulation has been implemented for the performance evaluation of SCPN models so far. A recent improvement are transient and steady-state SCPN simulations with statisical error control and on-line graphical output of results.

In addition to a standard simulation, a model-distributed simulation method has been implemented as a prototype. This module allows the efficient optimistic simulation of complex models on a cluster of workstations, exploiting fine-grained model partitioning and optimized rollbacks based on a special notion of concurrent time [34].

Rare-event simulation of SCPN models with the RESTART method has been proposed in [6], [35], and is now implemented in TimeNET 4.1. This allows evaluating the reliability of complex safety-critical systems.

The *token game*, i.e., an interactive simulation of the behavior of a model, is available for eDSPNs and SCPNs in TimeNET. Enabled transitions are highlighted in the graphical user interface, and are fired by clicking them with the mouse. The state is updated and shown in the window. A recent extension is an automatic animation of the model behavior for eDSPNs.

B. User Interface and Tool Use

The graphical user interface (GUI) of TimeNET had been completely rewritten in JAVA for version 4. It is generic in the sense that more or less any graph-like modeling formalism can be easily integrated without much programming effort. This is achieved using a XML description of net classes that are not only used to implicitly define the model file format(s), but also all drawing elements for the GUI as well as the icons that are shown to the user. The GUI is thus not restricted to Petri nets. As a stand-alone program it is named PENG, which is short for *platform-independent editor for net graphs*.

Model classes are described in an XML schema file, which defines the elements of the model. Node objects, connectors and miscellaneous others are possible elements. Nodes can be hierarchically refined by corresponding submodels, which may even belong to a different net class. For each node and arc type of the model the corresponding attributes and the

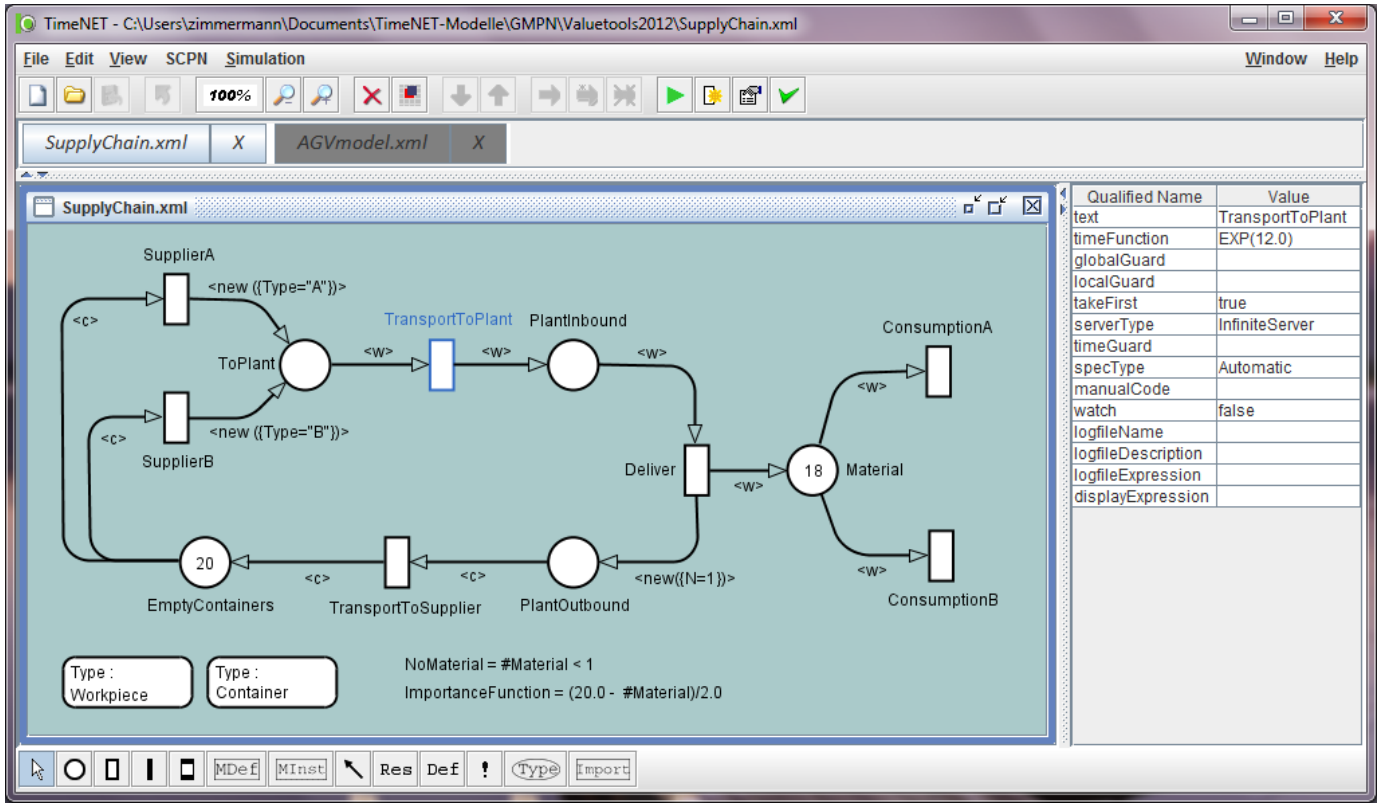

Fig. 1. Sample screen shot of TimeNET's graphical user interface.

graphical appearance are specified. The shape of each node and arc is defined using a set of primitives (e.g. polyline, ellipse, and text). Shapes can depend on the attribute value of an object, making it possible to show, e.g., tokens as dots inside places. Actual models are stored in an XML file that must be consistent with the model class definition, which can be checked automatically with library toolkits for XML. Editing and storing a model can already be done after the corresponding schema is available.

Figure 1 shows a sample screen shot of the GUI during an editing session of a SCPN model of a logistic process. There are standard menus with the necessary editing commands in the top row, e.g., **File** to open, close, save, create a model, or to export it in SVG format for printing. Commands under the entries **Edit** (cut, copy, paste, undo, ...) and **View** (grid on/off, zoom, go up or down in the hierarchy, ...), **Window**, and **Help** should be self-explanatory and follow common GUI style. There is a set of icons below the menu bar where the modeler can access menu commands which are most commonly used. Below this are buttons to switch between multiple windows in a tab-like fashion since version 4.1.

Program modules implementing model-specific algorithms can be added to the tool. A module has a predefined interface to the main GUI program. It can select its applicable net classes and extend the menu structure by adding new algorithms. All currently available and future extensions of net classes and their corresponding analysis algorithms are thus integrated with the same "look-and-feel" for the user, and reuse of existing program parts for other net classes is simplified.

For the eDSPN net class, as an example, the menu bar is extended by **Validation** (including commands for state-space size estimation; token game; and structural analysis covering P and T semi flows, extended conflict sets, traps, as well as siphons), **Evaluation** (with numerous simulation and analysis methods and variants, such as stationary analysis, stationary simulation, RESTART simulation, transient analysis, and transient simulation), and **Export** with the possibility to export the current model into the former .TN file format as well as the PNML exchange format. The **File / Open** command reads eDSPNs in .XML, .TN, and .pnml format.

The main window contains the drawing area. Models can be edited with the left mouse button like using a standard drawing tool with operations for selecting, moving, and others. The lower icon bar shows all model elements available in the current model class. The contents of this bar are automatically derived from the model class description. Clicking one of them changes the mouse pointer into a tool that creates the corresponding object. Arcs are added by clicking and holding at the source element, and then drawing the mouse to the destination.

Attributes of an element are edited by selecting the latter in the drawing area and then changing the values in the right tab. There is one entry in this window for every attribute as defined in the model class for that object. Transition

`TransportToPlant` is currently selected in Fig. 1, and the attributes of a SCPN transition are shown.

The dynamic behavior of a Petri net model can be checked with an interactive simulation, the token game. A recent extension is a non-interactive animation mode for eDSPN models, which will be available for SCPNs in the near future as well. A background simulation without a visualization can be started for SCPN models, for instance. Once the actual simulation program is created and running, the *result monitor* is started as well. This JAVA program receives all result measures from the simulation during run time and displays them graphically in windows. Moreover, it allows to inspect the final results in various ways. A sample screen shot is shown in the example section (Fig. 4).

C. Software Architecture

The main constituents of TimeNET are the graphical user interface (GUI) and analysis algorithms. The latter are usually started as background processes from the GUI, but can be run from the command line prompt as well. Shell scripts control the execution. Data exchange between GUI and analysis algorithms is mainly done with data files, while sockets are used between processes for efficiency.

A major goal in the development of TimeNET 4 was platform-independency, to allow Windows users access to the tool. The GUI has thus been completely rewritten in JAVA, while the internal evaluation modules are primarily implemented using C++ (older parts in C) for a better computational efficiency. The whole tool now runs on both Unix- and Windows-based environments, although being compiled from the same source tree. This has been achieved by using development tools and run-time libraries that can be configured on both platforms, such as Eclipse and the gcc compiler. Only shell scripts are written in their individual platform-specific language. For the rest of platform dependencies, a few switch macros or statements covering incompatibilities are sufficient.

Software development for TimeNET is usually done by students as part of their Master or Ph.D. thesis. It is thus of high importance to keep all analysis components modular with well-defined interfaces. The overall tool architecture as well as the graphical user interface have to be extendable and adaptable to new net classes and analysis algorithms, to allow easy prototypical implementations of new research ideas.

Figure 2 shows the interaction between GUI and other programs for the simulation of stochastic colored Petri nets. The model is edited with the GUI according to the SCPN model class. Models as well as model class descriptions are stored in XML format. The tool architecture allows to run the graphical user interface on a client desktop PC, while the computationally expensive simulations run on a remote server. Both parts may reside on the same host as the desktop default. This aspect is implemented with a simple middleware-like *remote system*. It allows to start and stop programs, transfer input and output data, and other functions independently of whether the interacting programs are located on machines running Unix or Windows.

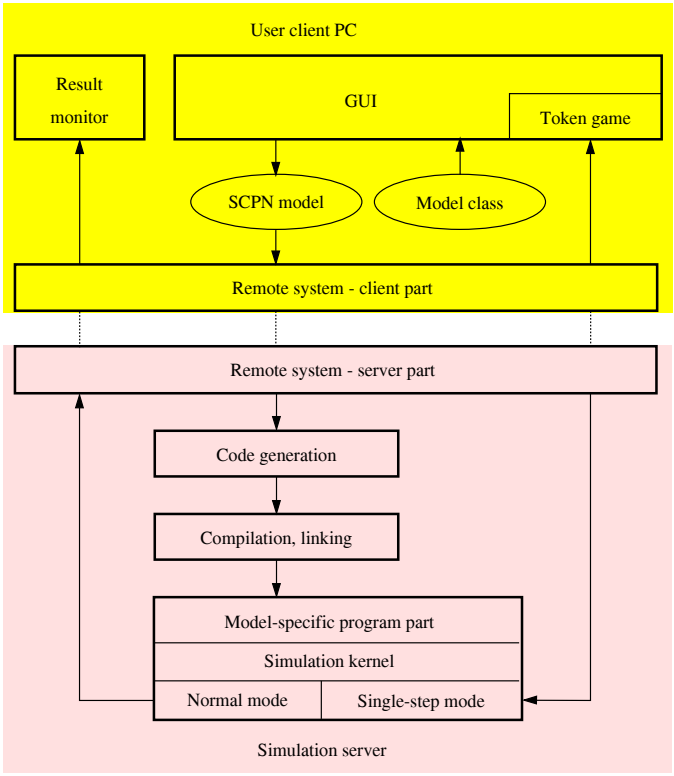

Fig. 2. Software architecture for SCPN modeling and evaluation

The program code implementing the SCPN transition's functionality is generated at the beginning of a simulation run by an automatic code generation module. This takes the model as input, and writes C++ classes for each token type and transition that has been changed since the last simulation. The resulting code is then compiled forming a model-specific program, which is linked to a simulation kernel to generate the actual simulation program. By doing so, efficiency of the later simulation program increases substantially compared to a standard simulation which "interprets" the model parts during the run. While such an approach may work well for uncolored nets (where tokens in places are just numbers), it turns out to be quite complex with the set of possibilities of a SCPN. In fact, the actual program code that checks for enabling of transition bindings, or fires the transition is completely encoded in the generated C++-code. Code generation and linking is done on the remote simulation server. If necessary, a data base can be combined with the tool to store model information. A SCPN model can thus be configured using data from the data base, using the ODBC standard as interface.

The simulation program has two working modes: normal simulation, which is intended for an efficient computation of performance measures; and a single-step mode used in conjunction with the GUI for an interactive visualization of the behavior (token game). Results of the simulation run are graphically displayed on the client PC during the simulation in a result monitor program. They are also stored in files that can be analyzed after a completed simulation.

Performance evaluation modules for eDSPNs do not use the remote system. They usually run on the client PC, while time-consuming evaluation tasks such as replications of simulation runs as well as transient solution of the Markov chains associated with non-exponential transitions in a stationary eDSPN solution can be done in parallel using several (Linux) PCs.

D. Application Areas

TimeNET has been used in a variety of areas, including

- Communication systems [36], [23],
- Manufacturing systems [26] (as well as their indirect optimization [31]),
- Reliable train control [30],
- Dependability [37], [38],
- Business Processes [39],
- Logistics and supply chains [40], [33], and
- Electronic funds transfer systems [41].

A more comprehensive list of references is given in [6].

III. What is new in TimeNET 4.1

This section lists some of the improvements that have been implemented since 2006, when version 4.0 Beta became available to the public [5].

For the eDSPN net class, the following features are new:

- The structural analysis module has been extended and derives and displays T semiflows (informally, similar to transition invariants).
- Probabilities of token distributions in the places of an eDSPN model can now be displayed (numbers and graphics) after an analytical solution.
- The token game has been significantly extended and can now be run in a non-interactive mode with adjustable speed to show a visualization of the net behavior. The user can choose if transition firing times should be used to control firing speed or if the firing should be done in a clocked fashion.
- The result monitor, which was originally implemented for the SCPN model class, is now fully integrated with eDSPN models as well. Results of a transient analysis or simulation are shown during run-time showing the evolving curves, as well as afterwards displaying the final results. Values of performance measures for a set of automatically executed solutions (denoted *experiment* in the tool) can be visualized versus the changing parameter.
- eDSPN models can be exported and imported in the *Petri net markup language* (PNML) exchange format.
- Especially for situations in which models are automatically generated and TimeNET's analysis modules are used as a part of another tool chain, the allowed string lengths for names (model object identifiers) as well as for measure expressions have been extended substantially.
- A graphical visualization of the reachability graph is currently under development.

Added functions for SCPN models include:

- Transition delays can now be marking-dependent as well as color-selective.
- Allowed delay distributions have been extended by uniform (`Uni` and `DUni`), triangular (`Triang`), and normal (`Norm`).
- Performance measures can be defined for individual colors (token types) in place markings and transition firings.
- Compared to the simple one-shot transient simulation available before, the tool now includes regular transient and steady-state simulation methods with statistical accuracy control (confidence interval, relative error).
- All simulation results are visualized in the result monitor, including confidence intervals.
- Rare-event simulation has been added for SCPNs.
- Implementation of the extended token game functionality available for eDSPNs is under way for SCPNs.

Some technical improvements for graphical user interface and the tool in general:

- A self-contained installation program is available for the Windows environment (a zip-archive based installation is still possible).
- TimeNET has been ported to Windows 7.
- Multiple windows containing models are now arranged in a tabbed layout to make switching between them easier.
- Curves and graphics displayed in the result monitor can now be exported as a SVG image for editing and inclusion in other documents.
- A help system based on the user manual has been implemented.
- Numerous bugs in all modules have been solved.
- A set of example models with descriptions has been put together for eDSPN and SCPN models.

TimeNET is an academic tool, and used for prototype development in research projects. Thus there are some additional modules which are not in a finished or documented state for the public. Examples are the model class stochastic UML state charts as well as distributed simulation techniques for SCPN models in a cluster of workstations.

IV. An SCPN Example

This section demonstrates some recently added features of TimeNET. As a demonstration example we introduce a SCPN model of a flexible manufacturing system (FMS) with an automated guided vehicle (AGV). Manufacturing systems are among the classic application areas of Petri nets, and many results and applications have been reported in the literature (compare [42] for a survey and bibliography).

The example considered in the following has been introduced in [1] (Section 8.3.2) with a GSPN model. Parts are transported throughout the FMS mounted on type-specific pallets. A Load/Unload station (LU) unmounts finished parts (which don't have to be considered further in the model), and puts a new part on the pallet. A *push* policy is adopted, meaning that there are always raw parts available and the throughput of the system is only limited by the speed of machines and transport system.

Furthermore, there are three machines (M1, M2 and M3) available, and one AGV which is responsible for all transport tasks between LU and the machines.

Two types of product are considered, named A and B. The work plans include alternative schedules and are set as follows: Parts of type A are processed by M1 first (processing time 10 seconds), and either M2 or M3 afterwards (15s or 10s, respectively). Type B parts only go through either M2 (20s) or M3 (30s). All parts have to be transported back to LU after being finished by M2 or M3. Loading/unloading takes 4s, while each AGV transport requires 2s time. A scheduling decision is necessary at M2/M3 because of the alternative work plans. In this paper the same policy is chosen as in the GSPN model in Figure 8.12 of [1]: To enforce machines to process the "faster" part, parts of type A have priority at M3, while B parts are preferred at M2. In case a machine is free and there is no prioritized part waiting, a part of the other type is selected.[1]

Performance evaluation questions for such a system could analyse the overall throughput and the individual throughput per part as well as their dependence on the number of pallets. Other analyses may answer machine selection problems, bottleneck detection, or how a second AGV would improve the throughput.

The FMS example is modeled with a SCPN in TimeNET, the resulting model is shown in Figure 3.[2] The "second experiment" shown in Figure 8.12 of [1] has been selected. As a by-product, it shows how a colored model describes the same behavior in a more compact and generic way (i.e., the number of part types would not add much to the net structure, while it significantly increases GSPN net size). The original model has 25 places and 24 transitions compared to only 13 places and 16 transitions in the colored model presented here.

Two token types are user-defined in the model: `Part` with attribute `string name` denoting the type, and `AGV` with attribute `Part p` (denoting the part reference that is currently loaded onto the AGV). Theer are not specific pallet tokens, as they only exist with parts loaded onto them except for the short time during load/unload operation. Tokens are internally treated as C++ objects; thus, colors (or token types) are related to class definitions.

Input arc inscriptions (such as `x`) describe variables that are bound to tokens from the corresponding input place. The association of an available token to each of the input variables of a transition denotes a *binding*, which can be checked for being enabled and has an associated firing delay distributions (for timed transitions) or weight (for immediate ones). When a transition fires, tokens specified by the output arc inscriptions are added to the output places, by referring to the input tokens via variable names. For a full definition of the type of stochastic colored Petri nets used in TimeNET, the reader is referred to [6] or the TimeNET user manual.

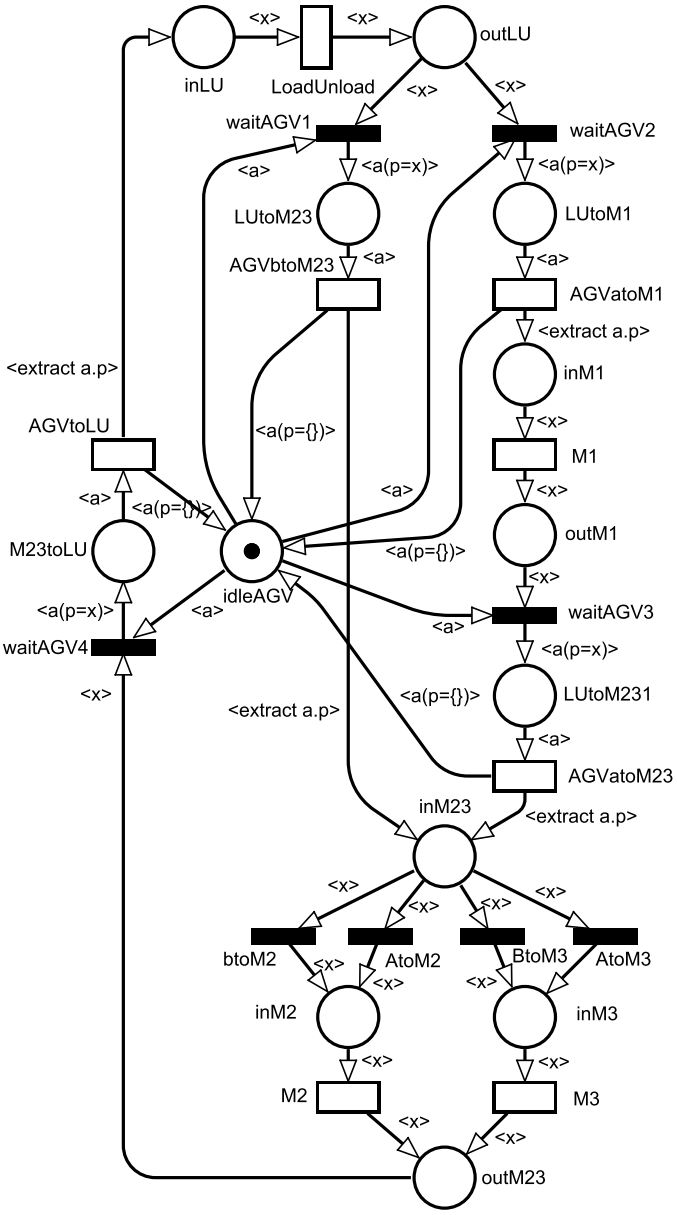

Fig. 3. Colored Petri net model of the example

After leaving the load/unload station mounted on a pallet (where the processing needs 4s), tokens have to be transported either to machine 1 or to machines 2 and 3. Both tasks are done by the AGV, which is modeled in the usual semaphore-like way (`idleAGV` and for each transport task a sequence of start transition `waitAGV*`, in-transport place `*to*`, and transport delay transition `AGVto*` (2s each). When an AGV task starts, a reference to the transported part token is stored in the AGV token (`a(p=x)`). The reference is transformed back into a token when the transport has finished (`extract a.p`), while the AGV token is emptied (`a(p={})`). Scheduling policies for the AGV can be implemented using transition priorities. For the first transport after the load/unload station, the destination

[1]This behavior differs slightly from the text in [1], but reproduces the behavior modeled in the figure.

[2]The model has been exported in SVG format and transformed into a PDF (using Inkscape). No post-processing was necessary except for removing type (color) definitions and result measure objects to save some space.

Measure	TimeNET	Result in [1]	Error
ThroughputAll	0.113838	0.107327	6.1%
ThroughputA	0.040232	0.042504	5.3%
ThroughputB	0.059607	0.064823	8.0%

TABLE I
COMPARISON OF SIMULATION RESULTS

depends on their part type. This is simply modeled with guard functions associated to `waitAGV1` and `waitAGV2`, e.g., `x.name=="B"`. Thus a binding can be enabled only if the type of product is of type B.

Machines are modeled by their input and output buffers (e.g., places `inM2` and `outM2` for machine 2) and a transition representing the processing delay (e.g., `M2`). In most cases simply the exponential delay distribution with firing delay as given in [1] is used, except for the more complex machine delays. As an example, Machine 2 has a different firing delay for the two part types, which is modeled by a color- and marking-dependent expression `EXP(15*#inM2(t.name=="A") +20*#inM2(t.name=="B")` describing an exponential distribution with mean delay of either 15 or 20, depending on the current part.[3] The restriction that each machine can process only one part at a time is simply done by using an *exclusive server* semantics for all machine-delay transitions. For M2 and M3, the corresponding input places `inM2` and `inM3` have a restricted capacity of 1 each to ensure scheduling decisions at the time when a machine becomes empty only. The decision about which part to process at machines M2 and M3 is done with the four transitions `btoM2`, `AtoM2`, `BtoM3`, and `AtoM3`. Using transition priorities and guard functions, the desired behavior is specified.

For the initial marking, one AGV token is put in place `idleAGV`, while as many pallet/part tokens as chosen for each type A or B are put into place `inLU` (not shown in the figure).

The model is then evaluated with a steady-state simulation using TimeNET. To compare our results with the ones given in [1], the same settings as for the second experiment are used, in which the influence of pallet numbers for both part types on the throughput values are evaluated. We arbitrarily chose $N = 3$ pallets for type A, and $M = 4$ for type B as done in Table 8.7 of the reference.

Performance measures are added to the model for the overall throughput (`ThroughputAll = #AGVtoLU`) and the throughput for the specific parts (e.g., for A: `ThroughputA = #AGVtoLU(a.p.name=="A")`). The mean number of tokens of type A in machine 2, for instance, can be measured with `AinM2 = #inM2(t.name=="A")` as an example of a measure using a place marking. In the case of a place, the #-sign denotes the number of tokens in it, while for transitions it means the number of firings. Both types of measures can be made color-specific by adding an expression in parentheses.

The results of the simulation are shown in Table I, compared with the analytical results from Table 8.7 of the reference. The experiment was done on a laptop computer with an Intel Core i5 M520 at 2.4 GHz in a 64bit Windows 7 environment. Full code generation and compilation of the complete model took 30s (less if only parts of the model are changed), while the actual simulation run time was less than one second to achieve a statistical accuracy of 5% relative error for a confidence interval size corresponding to 95%. Accuracy-controlled steady-state and transient simulations are new in TimeNET 4.1, just like the possibility to visualize mean transient behavior of performance measures together with their confidence intervals as shown in Figure 4 for the overall throughput.

[3]Such a delay could be expressed more elegantly using an input-variable `x` depending delay, which is a feature currently under development.

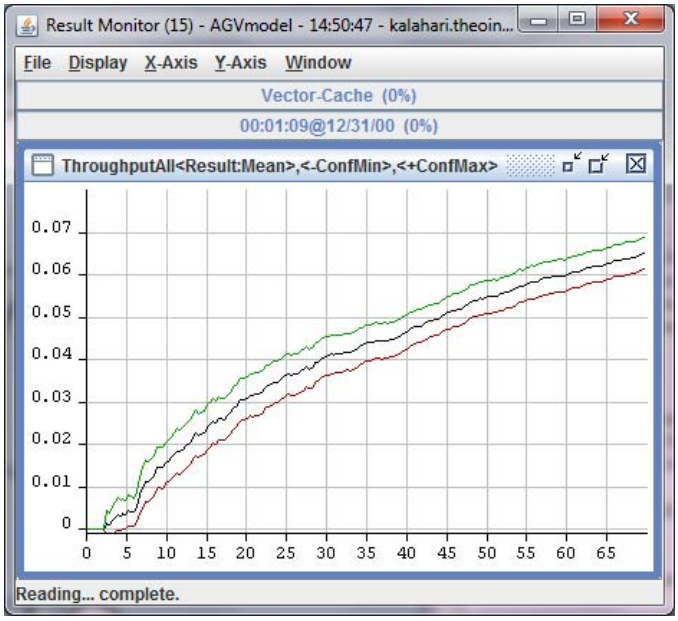

Fig. 4. Result monitor showing the transient evolution of overall throughput with confidence intervals

V. CONCLUSIONS

This paper introduced version 4.1 of the software package TimeNET, a tool for the modeling and performance evaluation with different types of stochastic Petri nets. Among the new features for stochastic colored Petri nets are stationary and transient simulation with statistical accuracy control, extended color-dependencies for their inscriptions, and rare-event simulation. Improvements for (uncolored) extended deterministic and stochastic Petri nets include a non-interactive token game animation, better graphical display of analysis results, and PNML import and export. A self-contained installation program as well as a help system have been implemented, and the whole tool is now available for Windows 7.

Future plans for further improvements include the semi-automatic parameter selection for rare event simulation; indirect optimization of complex system models with accuracy-adaptive simulation; domain-specific modeling and evaluation for energy-efficient embedded systems; implementation of a net class and analysis modules for stochastic automata; and a port to a native 64 bit application.

VI. Acknowledgments

The author wishes to thank the numerous colleagues and students who have actively contributed to the development of TimeNET over the years. The full list is much too long for this paper, but maintained in the user manual. Contributions to the recent improvements have been implemented by Marcel Schönfeld, Clemens Tirsch, Timur Ametov, Paula Herber, Sascha Kühne, Pawan Pokhrel, Solomon Oyelere, and Alexander Glatho.

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, ser. Series in parallel computing. John Wiley and Sons, 1995.

[2] L. M. Hillah, E. Kindler, F. Kordon, L. Petrucci, and N. Trves, "The Petri net markup language and ISO/IEC 15909-2," in *Proc. 10th Int. Workshop on Practical Use of Colored Petri Nets and the CPN Tools – CPN'09*. Aarhus University, 2009.

[3] C. Lindemann, *Performance Modelling with Deterministic and Stochastic Petri Nets*. Wiley, 1998.

[4] A. Zimmermann, J. Freiheit, R. German, and G. Hommel, "Petri net modeling and performability evaluation with TimeNET 3.0," in *Proc. of the 11th Int. Conf. on Tools and Techniques for Computer Performance Evaluation*, Schaumburg, Illinois, USA, 2000, pp. 188–202.

[5] A. Zimmermann, M. Knoke, A. Huck, and G. Hommel, "Towards version 4.0 of TimeNET," in *13th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems (MMB 2006)*, March 2006, pp. 477–480.

[6] A. Zimmermann, *Stochastic Discrete Event Systems*. Springer, Berlin Heidelberg New York, 2007.

[7] B. R. Haverkort and I. G. Niemegeers, "Performability modelling tools and techniques," *Performance Evaluation*, vol. 25, no. 1, pp. 17–40, 1996.

[8] K. S. Trivedi, B. R. Haverkort, A. Rindos, and V. Mainkar, "Techniques and tools for reliability and performance evaluation: problems and perspectives," in *Proc. 7th Int. Conf. on Computer performance evaluation: modelling techniques and tools*. Vienna, Austria: Springer Verlag, 1994, pp. 1–24.

[9] Petri net home page, http://www.informatik.uni-hamburg.de/TGI/PetriNets.

[10] S. Bernardi, C. Bertoncello, S. Donatelli, G. Franceschinis, G. Gaeta, M. Gribaudo, and A. Horvàth, "GreatSPN in the new millenium." Universität Dortmund (Germany), Research Report 760/2001, Sep. 2001, tools of Aachen 2001, Int. Multiconf. on Measurement, Modelling and Evaluation of Computer-Communication Systems, pages 17–23.

[11] P. Bonet, C. M. Llado, R. Puijaner, and W. J. Knottenbelt, "PIPE v2.5: A Petri net tool for performance modelling," in *Proc. 23rd Latin American Conference on Informatics (CLEI 2007)*, San Jose, Costa Rica, oct 2007.

[12] C. Hirel, B. Tuffin, and K. S. Trivedi, "SPNP: Stochastic Petri nets. version 6.0." in *Computer Performance Evaluation, Modelling Techniques and Tools — 11th Int. Conf., TOOLS 2000*, ser. Lecture Notes in Computer Science, vol. 1786. Schaumburg, IL, USA: Springer Verlag, 2000, pp. 354–357.

[13] A. Bobbio, A. Puliafito, M. Scarpa, and M. Telek, "WebSPN: Non markovian stochastic Petri net tool," in *Proc. 18th Int. Conf. on Application and Theory of Petri Nets*, Toulouse, France, Jun. 1997, pp. 24–33.

[14] G. Bucci, L. Carnevali, L. Ridi, and E. Vicario, "Oris: a tool for modeling, verification and evaluation of real-time systems," *Int. Journal of Software Tools for Technology Transfer*, vol. 12, no. 5, pp. 391–403, 2010.

[15] K. Jensen and L. M. Kristensen, *Coloured Petri Nets*. Springer-Verlag Berlin Heidelberg, 2009.

[16] K. Jensen, K. L. Kristensen, and L. Wells, "Coloured Petri nets and CPN tools for modelling and validation of concurrent systems," *Int. Journal on Software Tools for Technology Transfer (STTT)*, vol. 9, no. 3–4, pp. 213–254, 2007.

[17] C. Hirel, R. A. Sahner, X. Zang, and K. S. Trivedi, "Reliability and performability modeling using SHARPE 2000," in *11th Int. Conf. Computer Performance Evaluation: Modelling Techniques and Tools (TOOLS 2000)*, ser. Lecture Notes in Computer Science, B. R. Haverkort, H. C. Bohnenkamp, and C. U. Smith, Eds., vol. 1786. Schaumburg, IL, USA: Springer, 2000, pp. 345–349.

[18] M. Walter, M. Siegle, and A. Bode, "OpenSESAME — the simple but extensive, structured availability modeling environment," *Reliability Engineering and System Safety*, vol. 93, no. 6, pp. 857–873, 2008.

[19] C. Rohr, W. Marwan, and M. Heiner, "Snoopy - a unifying Petri net framework to investigate biomolecular networks," *Bioinformatics*, vol. 26, no. 7, pp. 974–975, 2010.

[20] F. Bause and P. Kemper, "QPN-tool for the qualitative and quantitative analysis of queueing Petri nets," in *Quantitative Evaluation of Computing and Communication Systems*. Springer Verlag, 1994, vol. 794, pp. 224–238.

[21] S. Spinner, S. Kounev, and P. Meier, "Stochastic modeling and analysis using QPME: Queueing petri net modeling environment v2.0," in *Proceedings of the 33rd International Conference on Application and Theory of Petri Nets and Concurrency (Petri Nets 2012)*, ser. Lecture Notes in Computer Science. Springer, June 2012.

[22] T. Courtney, S. Gaonkar, K. Keefe, E. W. D. Rozier, and W. H. Sanders, "Mbius 2.3: An extensible tool for dependability, security, and performance evaluation of large and complex system models." in *Proc. 39th IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN 2009)*, Estoril, Lisbon, Portugal, Jun. 2009, pp. 353–358.

[23] R. German, *Performance Analysis of Communication Systems, Modeling with Non-Markovian Stochastic Petri Nets*. John Wiley and Sons, 2000.

[24] R. German and J. Mitzlaff, "Transient analysis of deterministic and stochastic Petri nets with TimeNET," in *Proc. Joint Conf. 8th Int. Conf. on Modelling Techniques and Tools for Performance Evaluation*, ser. Lecture Notes in Computer Science. Springer Verlag, 1995, vol. 977, pp. 209–223.

[25] A. Heindl and R. German, "A fourth order algorithm with automatic stepsize control for the transient analysis of DSPNs," *IEEE Transactions on Software Engineering*, vol. 25, pp. 194–206, 1999.

[26] A. Zimmermann, J. Freiheit, and A. Huck, "A Petri net based design engine for manufacturing systems," *Int. Journal of Production Research, special issue on Modeling, Specification and Analysis of Manufacturing Systems*, vol. 39, no. 2, pp. 225–253, 2001.

[27] C. Kelling, "TimeNET$_{\text{sim}}$ – a parallel simulator for stochastic Petri nets," in *Proc. 28th Annual Simulation Symposium*, Phoenix, AZ, USA, 1995, pp. 250–258.

[28] M. Villén-Altamirano and J. Villén-Altamirano, "Analysis of RESTART simulation: Theoretical basis and sensitivity study," *European Transactions on Telecommunications*, vol. 13, no. 4, pp. 373–385, 2002.

[29] C. Kelling, "A framework for rare event simulation of stochastic Petri nets using RESTART," in *Proc. of the Winter Simulation Conference*, 1996, pp. 317–324.

[30] A. Zimmermann and G. Hommel, "Towards modeling and evaluation of ETCS real-time communication and operation," *Journal of Systems and Software*, vol. 77, pp. 47–54, 2005.

[31] A. Zimmermann, D. Rodriguez, and M. Silva, "A two phase optimisation method for Petri net models of manufacturing systems," *Journal of Intelligent Manufacturing*, vol. 12, no. 5/6, pp. 409–420, Oct. 2001, special issue "Global Optimization Meta-Heuristics for Industrial Systems Design and Management".

[32] J. Trowitzsch, D. Jerzynek, and A. Zimmermann, "A toolkit for performability evaluation based on stochastic UML state machines," in *Proc. 2nd Int. Conf. on Performance Evaluation Methodologies and Tools (Valuetools 2007)*, Nantes, France, Oct. 2007.

[33] A. Zimmermann, M. Knoke, S.-T. Yee, and J. D. Tew, "Model-based performance engineering of General Motors' vehicle supply chain," in *IEEE Int. Conf. on Systems, Man and Cybernetics (SMC 2007)*, Montreal, Canada, Oct. 2007, pp. 1415–1420.

[34] M. Knoke, F. Khling, A. Zimmermann, and G. Hommel, "Towards correct distributed simulation of high-level Petri nets with fine-grained partitioning," in *2nd Int. Symp. Parallel and Distributed Processing and Applications (ISPA'04)*, ser. Lecture Notes in Computer Science, J. Cao, Ed., vol. 3358. Hong Kong, China: Springer Verlag, Dec. 2004, pp. 64–74.

[35] A. Zimmermann, "RESTART simulation of colored stochastic Petri nets," in *Proc. 7th Int. Workshop on Rare Event Simulation (RESIM 2008)*, Rennes, France, Sep. 2008, pp. 143–152.

[36] A. Heindl and R. German, "Performance modeling of IEEE 802.11 wireless LANs with stochastic Petri nets," *Performance Evaluation*, vol. 44, pp. 139–164, 2001.

[37] A. Zimmermann, "Dependability evaluation of complex systems with TimeNET," in *Proc. Int. Workshop on Dynamic Aspects in Dependability Models for Fault-Tolerant Systems (DYADEM-FTS 2010)*, Valencia, Spain, Apr. 2010.

[38] A. Zimmermann and J. Trowitzsch, "Reliability evaluation of distributed embedded systems with UML state charts and rare event simulation," in *Tagungsband des Dagstuhl-Workshops MBEES: Modellbasierte Entwicklung eingebetteter Systeme*, H. Giese, Ed. Technische Universitt Braunschweig, 2009, pp. 128–139.

[39] J. Dehnert, J. Freiheit, and A. Zimmermann, "Modelling and evaluation of time aspects in business processes," *Journal of the Operational Research Society*, vol. 53, pp. 1038–1047, 2002.

[40] G. Alves, P. Maciel, P. Lima, R. Massa Ferreira, and A. Zimmermann, "Automatic modeling for performance evaluation of inventory and outbound distribution," *IEEE Trans. on Systems, Man and Cybernetics. Part A, Systems and Humans*, vol. 40, no. 5, pp. 1025–1044, 2010.

[41] C. Araújo, P. Maciel, A. Zimmermann, E. Andrade, E. Sousa, G. Callou, and P. Cunha, "Performability modeling of electronic funds transfer systems," *Computing*, vol. 91, no. 4, pp. 315–334, 2010, springer Wien.

[42] M. Zhou and K. Venkatesh, *Modeling, Simulation, and Control of Flexible Manufacturing Systems*, ser. Intelligent Control and Intelligent Automation. World Scientific, 1999.