A PETRI NET APPROACH TO ARGESIM COMPARISON C2 'FLEXIBLE ASSEMBLY SYSTEM' USING THE MATLAB PETRISIMM TOOLBOX

Thomas Löscher[(a)], Felix Breitenecker[(b)]

[(a)]PROFACTOR GmbH, Austria
[(b)] Vienna Univ. of Technology, Austria

[(a)]thomas.loescher@profactor.at, [(b)] felix.breitenecker@tuwien.ac.at

ABSTRACT
This paper deals with the modelling, simulation and optimisation of a flexible assembly system. Timed and Coloured Petri nets are used to represent the systems behaviour. The so called MATLAB PetriSimM toolbox forms the basis for modelling and extends the benchmark performed by the ARGESIM comparisons.

Keywords: process simulation, production simulation, resources planning

1. INTRODUCTION

Petri nets offer the capability to model discrete event systems on a very low level (Petri 1962). Representing resource sharing problems, conflicts or describing complex control strategies are the results of the use of their basic properties. A lot of applications exist in the field of modelling and simulating flexible manufacturing systems by means of Petri nets (Zuberek 1999). For evaluation purposes and management ratios it is necessary to simulate over the time domain. For Petri nets many extensions exist to implement time to their basic structure and definition (Zuberek 1991; Bowden 2000). The complexity of flexible manufacturing systems can be easily growing very high. Therefore other extensions are required to ensure the capability of meaningful modelling. Petri nets are also growing very fast if the complexity is increasing. Further extensions like the introduction of token colours are needed (Bause and Kritzinger 2002). In this simple approach the colours are used to simplify the graphical representation of the Petri net.

This paper shows the modelling, simulation and optimisation of a flexible assembly system. Basically, this problem definition is made for an entity flow approach where each processing unit is modelled as entity of the system. Petri nets do not really offer this way of modelling because tokens are destroyed if they are not needed any more and they are created if they are needed. Therefore it is not possible to track tokens as entities during the simulation and no further properties or attributes can be given to them. The use of Petri nets is a benchmark to test and show the feasibility of their capabilities.

2. PROBLEM DEFINITION

2.1. ARGESIM Comparisons

Simulation News Europe (SNE) features a series on comparisons of simulation software (Breitenecker 2008). Based on simple, easily comprehensible models special features of modelling and experimentation within simulation languages, also with respect to an application area, are compared.

Features are, for instance: modelling technique, event handling, numerical integration, steady-state calculation, distribution fitting, parameter sweep, output analysis, animation, complex logic strategies, sub-models, macros, statistical features etc.

Up to now 19 comparisons have been defined in Simulation News Europe, the series will be continued. Furthermore, a special comparison of parallel simulation techniques has been defined.

2.2. Comparison C2 – Flexible Assembly System

The following example of a flexible assembly system has been chosen because it checks two important features of discrete event simulation tools:

- The possibility to define and combine sub-models.
- The method to describe complex control strategies.

The model consists of a number of almost identical sub-models of the following structure (figure 1):

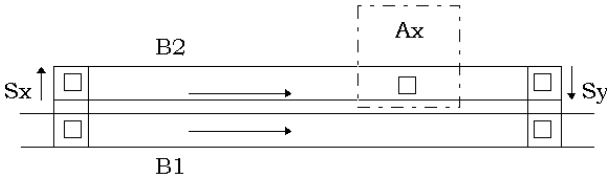

Figure 1: Submodel of assembly station

Two parallel conveyor belts, B1 and B2, are linked together at both ends. An assembly station Ax is placed at B2. Pallets are coming in on belt B1. If they are to be processed in Ax they are shifted in Sx to B2 and

possibly enter a queue in front of Ax. If there is no more empty buffer space on B2 or the pallet is not to be processed in Ax it continues its way along B1. Parts that have been processed in Ax are shifted back to B1 in Sy, having priority over those coming from the left on B1.

The total system now consists of 8 of these subsystems, varying in length, operation and operation time (see figure 2). Between two subsequent subsystems there is a space of 0.4 m, whereas pallets from the third subsystem A2 can be shifted directly to A3, and from A6 directly to A1. The shifting parts, however, cannot function as buffers, i.e. a pallet can only enter an Sx if it can leave it immediately.

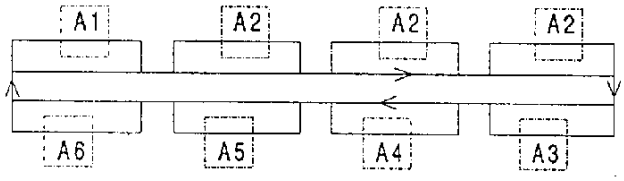

Figure 2: Total System

There are three identical stations A2 in the system, because the operation in A2 takes much longer than the other operations.

Unprocessed parts are put on pallets in A1. They can either be processed in A2 first, and then in A3, A4, A5, or in A3, A4, A5 first, and then in A2. The sequence of operations among A3, A4, and A5 is arbitrary. Station A6 is a substitute for any of the stations A3, A4, A5, i.e. whenever one of these stations is down, or the buffer in front of it is free, the corresponding operation can be executed in A6. Finished parts are unloaded in A1, unfinished parts enter another circle.

3. PETRISIMM TOOLBOX

The so called MATLAB PetriSimM toolbox is based on a basic toolbox (Mušič, Zupančič, and Matko 2003) which deals with analysis, supervisory control synthesis, and non-timed simulation. This basic toolbox is programmed in MATLAB version 5.3 and is therefore adapted to MATLAB version 7.2 (R2006a) to form the MATLAB PetriSimM toolbox. The toolbox is embedded in the MATLAB environment and its usage requires version 7.0 or higher. Furthermore the toolbox is extended with the capability of Timed Petri Nets and timed simulation using the holding durations principle (Löscher, Breitenecker, and Mušič 2005; Löscher, Breitenecker, Mušič, and Gradišar 2005; Löscher, Gradišar, Breitenecker, and Mušič 2006). In another step Coloured Petri Nets are developed for the use in the MATLAB PetriSimM toolbox. The enabling duration principle is added as a second approach of implementing time into Petri Nets. A new way of defining firing sequences is found to be able to model scheduling problems being independent of the occurrence of any conflicts (Löscher and Breitenecker 2006; Mušič, Löscher, and Gradišar 2006). Finally the toolbox is extended with the optimisation of scheduling problems containing heuristic algorithms like Simulated Annealing, Threshold Accepting and Genetic Algorithms (Löscher, Mušič, and Breitenecker 2007). In case of stochastic processes a sequential paired t-test and variance reduction techniques are used and implemented to solve the stochastic optimisation for sequencing and scheduling problems. To sum up, the sophisticated MATLAB PetriSimM toolbox offers the capabilities of analysis, modelling and simulation of Petri Nets. Furthermore it is possible to optimise scheduling problems based on Timed, Coloured, and Stochastic Petri Nets. The open source MATLAB PetriSimM toolbox can be used for education in a graduate level and for modeling and simulating real life processes of discrete event systems in equal measure.

3.1. GUI

Figure 3 shows a screenshot of the graphical user interface (GUI) of the PetriSimM toolbox. The GUI is divided into a menu bar, a button bar and an axes area. In the menu bar different modes can be chosen. The user can switch between analysis, non-timed simulation, timed simulation based on the holding durations principle, and timed simulation based on the enabling durations principle. Furthermore models can be saved, loaded, exported and printed. For each type of simulation several parameters can be set. Another important part of the menu bar is the options menu where the priority wizard and Gantt chart wizard can be started. Next to the options menu the optimisation menu is placed. There several parameters and modes for the optimisation can be changed and selected. The button bar contains several buttons for building, changing, zooming, simulating and analysing the Petri Net models. In the axes area the Petri Net models can be created, simulated and the so called token game can be shown. Figure 3 also contains the Petri Net model of a production cell which is later used for the optimisation.

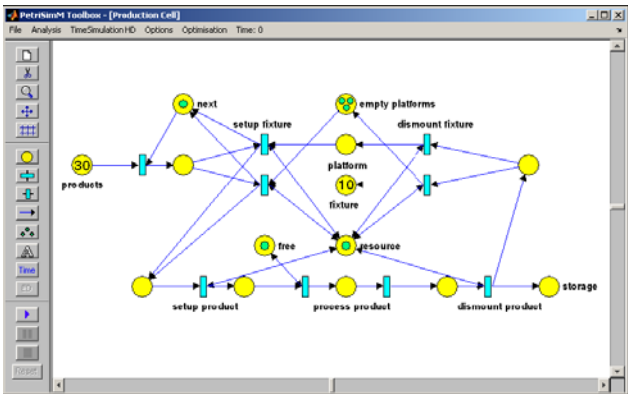

Figure 3: Graphical User Interface

3.2. Simulation

The simulation is a main part of the PetriSimM toolbox. It is separated into non-timed simulation, timed simulation using holding durations and timed simulation using enabling durations. For all three simulation modes an animation of the token game can be visualised. But this feature is only used for

educational purposes because through the animation the simulation speed is highly increased. It is possible to change to the animation speed in the parameter section and for the optimisation the animation is deactivated. Another interesting parameter for non-timed simulation is the firing probability. This parameter controls the firing of enabled transitions. This means that it is randomly decided if an enabled transition can □re or not depending on the defined firing probability. For timed simulation time delays can be assigned to the transitions which can be deterministic or stochastic. This means that any probability distribution function can be defined to each transition. For this purpose, any MATLAB m-file can be written resulting to a single positive value, or existing MATLAB probability distribution functions, which can be used to model stochastic time delays. Only the positive part of the used function is taken. If the result of the used stochastic function is negative the time delay is set to zero and a warning is displayed.

3.3. Priority Wizard

Petri nets offer the possibility to handle conditions on the highest level. This is a big advantage compared to the event-oriented tools for modelling and simulation of discrete systems. If conflicts or resource sharing problems occur, a strategy for solving simultaneous events should be available in these tools. Here, Petri nets realize these problems through the basic properties of their structure.

In the PetriSimM toolbox, a Priority wizard allows selecting groups of transitions to define a priority or a sequence for firing. These definitions are the basis for the resolution of conflicts. While in previous versions of the toolbox the firing sequences are dependent on the occurrence of a conflict, the sequence strategy in the current version is independent on conflicts. If a firing sequence is defined for a group of transitions no conflict of the involved transitions is required any more. Only the transition which is defined by the current value of the sequence vector will fire if it is enabled. All other transitions of the group are disabled until this specific transition has fired. This new strategy can lead to a new kind of deadlock if only the disabled transitions are enabled without consideration of the defined firing rules. This new approach offers the possibility to model queuing, sequencing or scheduling problems being independent of the appearance of any conflicts.

3.4. Data Handling

The allocation of the places over time is needed to show and to analyse results of the simulation. This means that the number of tokens has to be stored for each place and for each colour in each iteration step. The number of iterations depends on the specific problem. Therefore this value is not fixed and in general it can be very large. On this account very big matrices are created during the simulation. Dynamic memory allocation could be used to solve this problem. But the access to big matrices causes speed problems corresponding to the size of the used matrices. If the matrices and the stored information are increased too much memory problems can also occur. In the PetriSimM toolbox these problems are solved by the use of external binary files. The built-in MATLAB command fwrite is used to store the data during the simulation. At the end of the simulation the fread command is used to store the matrices in MATLAB as application data of the GUI. If the memory of the program is exceeded the binary files are transformed to text files in order that they can be used for post-processing using other tools or programming languages. The big advantage of this implementation is the fact that the data storing takes the same time in each iteration of the simulation algorithm. The used built-in functions are very fast and therefore only a marginal increase of computation time can be determined for the simulation function.

4. IMPLEMENTATION

A flexible assembly system described in the ARGESIM Comparison 2 (ARGE Simulation News 2008) is an atypical application for Petri Nets. The aim of this simulation and implementation was to prove that even such a process can be simulated by a discrete system based on Petri Nets, although the programming and processing efforts are considerably high. The C2 comparison "Flexible Assembly System" consists of two times two conveyors, eight assembly stations and two shifting parts.

4.1. Use of sub-models

The flexible assembly system is modelled with Petri nets using a discretization approach. The movements of the conveyor belts and the processing steps of the pallets are designed using three places units. A resource place ("free") signalizes whether entrance into the unit is possible or not (figure 4). During the given processing time the tokens pass from the entrance place to the exit place. This is modelled using the holding duration principle.

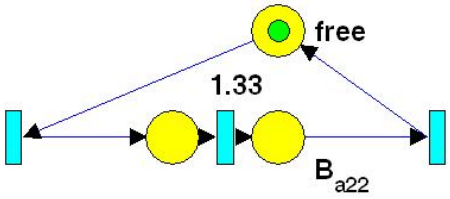

Figure 4: Conveyor unit

Figure 5 shows the corresponding processing unit. For a more concise data recording, the processing unit additionally disposes of a place ("processed") indicating when the pallet is assembled.

An assembly station (figure 6) consists of successive conveyor units and one processing unit, both mentioned above. The PetriSimM toolbox offers additionally the possibility to replicate existing structures and sub-nets. This feature simplifies the building and the creation of each assembly station and the whole flexible assembly system.

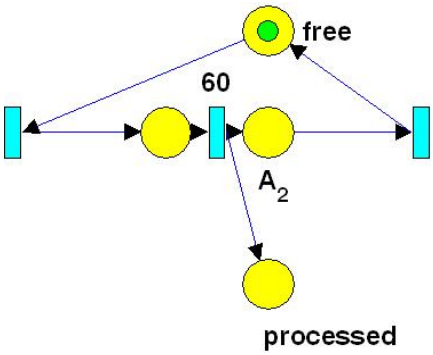

Figure 5: Processing unit

After building the basic structure of the net the connection logic (shifting and switching stations, etc.) of the sub-nets have to be established. Furthermore, a customizing of the different types of assembly stations has to be done.

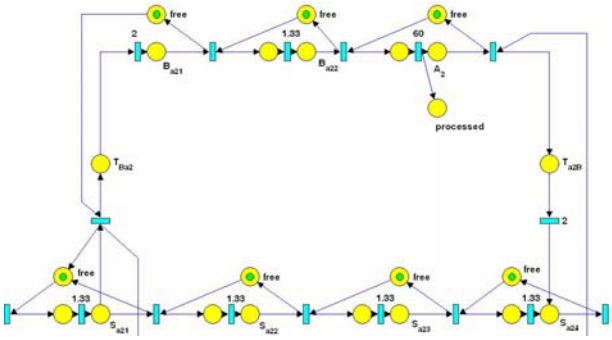
Figure 6: Assembly station

In order to enable simulations with a number of pallets exceeding the band capacity (40), an additional tool had to be introduced: this control unit permits loading of new pallets from the assembly stations to the main conveyor only in case of free capacity and thus avoids system deadlocks.

4.2. Prioritisation

The priorities at the switches in front of each assembly station have been assigned as follows:

1. Leaving the assembly station
2. Entering the assembly station
3. Forwarding along the conveyor

At the end of the conveyor, the shifting part obeys similar rules:

1. Shift directly from station to station
2. Shift from station to conveyor
3. Shift from conveyor to station
4. Shift from conveyor to conveyor

All these prioritisations can be represented as conflicts of the Petri Net. Therefore, priorities are assigned to selected transitions to solve these control strategies by the use of the basic properties of Petri Nets.

4.3. Use of colours

Figure 7 shows a screenshot of the Colour Wizard. The different colours represent the different types of tokens. In the parameter section the number of desired distinguishable tokens can be defined. If more than one token are chosen the so called Colour Wizard can be shown. The Colour Wizard can be used to change the look and the names of the different token colours. The colours are coded as RGB values and can be easily modified. It is also possible to reset all settings to the default values and names. The colours of the used different tokens are interesting for the animation of the simulation. The so called token game is mainly used for educational purposes.

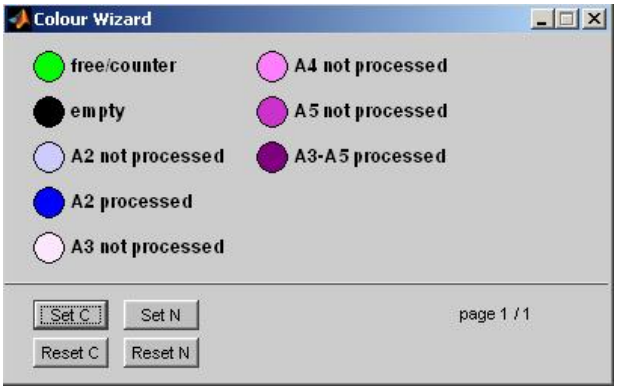

Figure 7: Colour Wizard

In most discrete simulators and discrete simulation languages the pallets could be modelled as entities. These entities could store information in several different attributes. Petri Nets offer entity simulation in a very rudimental way. Tokens could be used to represent entities of a system but there are not many possibilities to store information. A big disadvantage is the use of the tokens during the firing of transitions. Basically tokens are deleted if they are not needed any more and they are created if they are needed again. A track of information would be very complex and contradict to the definition of place/transition Petri nets.

In this case the colours of the tokens are used to model entities of the system. The pallets are modelled as combination of multiple coloured tokens, the colours making statements about the state of procession of the pallet. One pallet is represented by two coloured tokens. The different colours define the current processing status of the pallet (figure 7).

The Transition Wizard (figure 8) is used to modify and define the transitions in case of Coloured Petri Nets. The graphical description of Petri Nets can be simplified if many identical sub nets are used. Therefore the transitions of the sub nets are merged together and one transition of the Petri Net can represent and contain many other transitions. The Transition Wizard consists of several buttons corresponding to the following functionalities: transitions can be deleted, new transitions can be added, the names of the transitions can be modified and the weights of the corresponding arcs can be defined. Figure 8 shows a Transition Wizard

containing the transitions which controls one movement on the conveyor belt. For each transition different weights can be defined for the input and output arcs. These weights control and determine the way of each pallet through the system depending on its current status (colour and numbers of tokens).

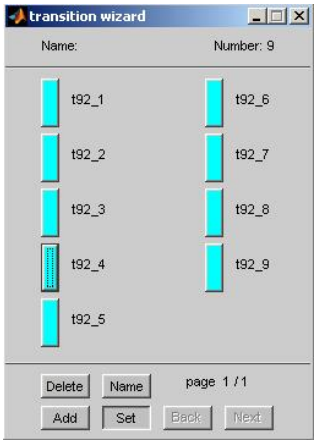

Figure 8: Transition Wizard

5. RESULTS

The complexity of the model (227 places, 157 transitions, 8 colours) results in large scaled input and output matrices (1816 x 929). Figure 9 shows a screenshot of the complete model.

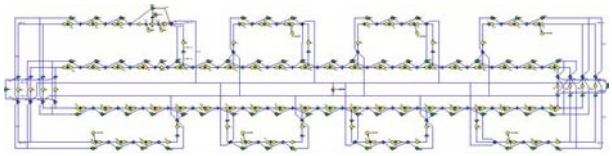
Figure 9: Model

This complexity increase both processing time and quantity of data. Each single simulation took between one and three days on a home computer. The allocation of all places are stored in external log files because the internal memory of MATLAB could not handle this big amount of data and information Furthermore, up to 29 GB of data were produced during one simulation session, in total 110 GB. The processing was therefore delegated to the phoenix-cluster of the University of Technology Vienna.

The evaluation of the ARGESIM comparisons is separated into several tasks. The following sections describe the tasks defined for comparison 2. Based on these results and tasks different simulators and simulation languages can be compared.

5.1. Control Strategy/Statistical Evaluation.

Due to systematic restrictions of the method and the model it is not possible to follow one single pallet throughout the process. The Petri net approach offers entities formed by a combination of coloured tokens but it is not possible to evaluate any statistical data during the simulation. Therefore the information of the allocation of certain places is stored over time in external log files and post-procession of this node data becomes necessary. Figure 10 shows a screenshot of the start and end point of the system. At the beginning and at the end tokens are created in the “in” and “out” places for each pallet, respectively. After the simulation it is not possible any more to get the throughput time for one single pallet but for statistical purposes the average throughput time can be easily calculated. This statistical evaluation was done by external post-processing of the log-files based on small external C programs. Following the recommendation, the data used was from the 120th to the 600th minute of simulation.

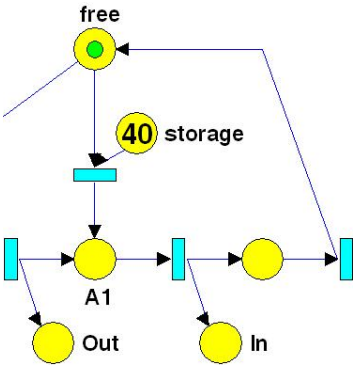

Figure 10: Start and end point

5.2. Simulation Results - Throughput.

The simulation time was eight hours. Table 1 show an extract of the throughput times and the graph of figure 11 aims to provide an overview of the system's behaviour.

Table 1: System behaviour: through put time (extract)

Number of pallets	Total throughput	Average throughput time [s]
5	480	292.50
10	960	292.50
13	1333	273.29
14	**1440**	**272.50**
15	1440	292.50
20	1444	391.92
40	1441	792.04
60	1433	1198.66

Larger calculations than with 40 pallets are purely theoretic, since they exceed the system's capacity. This was made possible by the deadlock control unit.

5.3. Simulation Results and Optimisation.

As shown in the following graph (figure 12), the utilization of stations A2–A6 reaches its possible maximum (100% utilisation of „bottleneck“ stations A2-1–A2-3) starting at 14 pallets. In this configuration, the highest possible throughput (1440 pallets; at some points slightly higher numbers may have encountered due to the delayed measurement) can be obtained at the lowest throughput time (272.50 seconds).

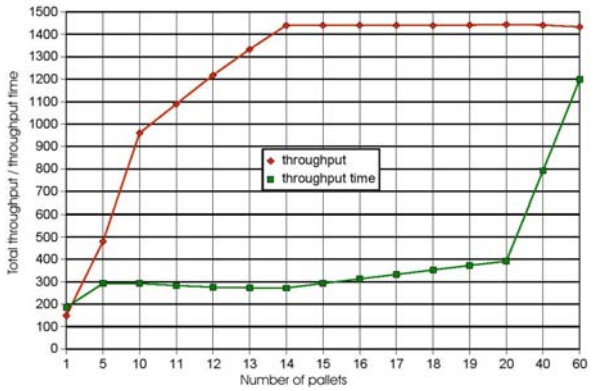

Figure 11: System behaviour – total throughput and throughput time

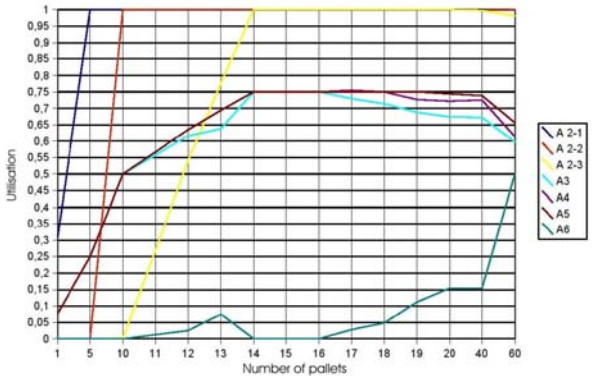

Figure 12: System behaviour: assembly stations' utilisation

6. CONCLUSIONS

The flexible assembly system can be modelled using the advantages and properties of Petri nets and the MATLAB PetriSimM toolbox. Although, the entity approach can not fully represented it is possible to fulfill all requirements of comparison 2 of the ARGESIM series.

REFERENCES

Petri C.A., 1962. *Kommunikation mit Automaten.* Dissertation, Universität Bonn.

Bowden F.D.J., 2000. A brief survey and synthesis of the roles of time in petri nets. *Mathematical and Computer Modelling* 31:55-68.

Zuberek W.M., 1991. Timed Petri Nets. Definitions, Properties and Applications. *Microelectronics and Reliability* 31(4):627-644.

Bause, F., Kritzinger P., 2002. *Stochastic Petri Nets - An Introduction to the Theory (2nd edition).* Germany: Vieweg Verlag.

Zuberek W.M., 1999. Timed Petri Nets in Modelling and Analysis of Simple Schedules for Manufacturing Cells. *Computers and Mathematics with Applications* 37:191-206.

Breitenecker F., 2008. *ARGESIM Benchmarks*. Available from: http://seth.asc.tuwien.ac.at/argesim/index.php?id=23 [accessed 12 July 2008]

Mušič G., Zupančič B., Matko, D., 2003. Petri Net Based Modelling and Supervisory Control Design in Matlab. *Proceedings of EUROCON Conference*, pp. 362-366. September 22-24, Ljubljana (Slovenia).

Löscher T., Breitenecker F., Mušič G., 2005. Petri Net Modelling and Simulation in Matlab - A Petri Net Toolbox. *Simulation News Europe* 43:20-21.

Löscher T., Breitenecker F., Mušič G., Gradišar D., 2005. A Matlab-based tool for timed Petri nets. *Proceedings of ERK Conference*, pp. 273-276. September 26-28, Portorož (Slovenia).

Löscher T., Gradišar D., Breitenecker F., Mušič G., 2006. Timed Petri Net Simulation in Matlab: A Production Cell Case Study. *Proceedings of MATHMOD Conf*erence, Proceedings on CD. February 7-10, Vienna (Austria).

Löscher T., Breitenecker F., 2006. Petri Net Modelling and Simulation of Production Processes with PetriSimM, a MATLAB-based Toolbox. *Proceedings of 12. ASIM - Fachtagung Simulation in Produktion und Logistik*, pp. 313-319. September 26-27, Kassel (Germany).

Mušič G., Löscher T, Gradišar D., 2006. An Open Petri Net Modelling and Analysis Environment in Matlab. *Proceedings of IMM Conference*, pp. 123-128. October 4-6, Barcelona (Spain).

Löscher T., Mušič G., Breitenecker F., 2007. Optimisation of Scheduling Problems based on Timed Petri Nets. *Proceedings of EUROSIM Conference*, Proceedings on CD. September 9-13, Ljubljana (Slovenia).

AUTHORS BIOGRAPHY

Thomas Löscher studied Technical Mathematics at the University of Technology of Vienna. During his diploma study he specialised in modelling, simulation and optimisation of discrete event systems. In his PhD study he continued his research in the field of simulation-based optimisation. He studied one semester abroad based on cooperation with the faculty of electrical engineering of the university Ljubljana, Slovenia. There he found the basis for his PhD thesis where he optimised scheduling problems based on Timed Petri nets. During his PhD study he also worked for the ARC Seibersdorf research company in the field of discrete event simulation. Currently he is employed as research associate at the Profactor GmbH in the field of simulation-based design and optimisation.