A New Petri Nets Based Approach for Modeling of Discrete Manufacturing System

Reggie Davidrajuh(✉)

Department of Electrical and Computer Engineering, University of Stavanger, Stavanger, Norway
Reggie.Davidrajuh@uis.no

Abstract. This paper presents a new approach for modeling discrete manufacturing systems using Petri nets. Starting with the literature review of the classical approaches, this paper describes the new approach known as Activity-Oriented Petri Nets (AOPN). The AOPN approach provides expressive Petri net models that are powerful enough for modeling complex and large-scale manufacturing systems. The AOPN approach provides compact Petri net model too when a large number of resources are used in the manufacturing system. GPenSIM is a software implementation of AOPN on MATLAB platform. Using a flexible manufacturing system as a running example, this paper also shows how a real-life manufacturing system can be modeled and simulated with the GPenSIM software using AOPN approach.

Keywords: Modeling of manufacturing systems · Discrete event systems · Petri nets · Activity-Oriented Petri Nets (AOPN) · GPenSIM

1 Introduction

Modern manufacturing systems are intelligent, complex, large, and expensive. For modeling these sophisticated systems, we need advanced tools. This paper focuses on discrete manufacturing systems and presents a new approach that is based on Petri nets.

Petri nets, because of its graphical nature, are self-documenting and easy to understand. The mathematics behind Petri nets is also easy to grasp. Hence, Petri nets are a suitable framework for modeling discrete event systems [1]. However, there is indeterminism in Petri nets, and it has a tendency to become huge when many resources are involved in the system.

This paper introduces a new approach known as the Activity-Oriented Petri Nets (AOPN) that provides a solution by developing discrete models in two phases: in the first phase, a simpler static Petri net model ("skeleton") is developed, and in the second phase, the run-time details are added. This approach, while keeping the Petri net model size to a minimum allows implementation of sufficient logic on the model so that advanced discrete manufacturing systems can also be modeled.

In this paper: A short literature review is given in Section 2. Section 3 presents the classical approach for modeling manufacturing systems with Petri nets. Section 4 presents the new approach AOPN. Finally, Section 5 presents some discussions.

A. Burduk and D. Mazurkiewicz (eds.), *Intelligent Systems in Production Engineering and Maintenance – ISPEM 2017*, Advances in Intelligent Systems and Computing 637,
DOI 10.1007/978-3-319-64465-3_11

2 Literature Review: The Subsystems of a Manufacturing System

According to [1], a manufacturing system is composed of two main parts: The physical system and the decision-making system (DMS).

2.1 The Physical System

The physical system can be divided into subsystems [2]:

- The active systems (e.g. machines).
- The passive systems (e.g. buffers).

Some of the physical systems can be active or passive (e.g. robots) depending on the situation. For instance, a robot can be used as an active system (machining an input material), whereas in some other situations, it can be used as a passive system (handling). Since this paper focuses only on modeling with Petri nets, the physical system is modeled as a decision-free Petri net, consisting of many modules that represent the functional entities (e.g. the machines, transportation systems, buffers, etc.). When modeling manufacturing systems with Petri nets, there is a degree of indeterminism involved in the model. For example, which transition is to fire if there are several transitions enabled by the same enabling tokens? In addition, Petri nets are incapable of realizing all the modeling logic of real-life manufacturing systems, despite some Petri nets extensions can do so at the expense of the analytical power [3].

2.2 The Decision-Making System (DMS)

The decision-making system (DMS) is to cope with the incapability of the physical system to make decisions on its own and to manage the indeterminism in the Petri net model. The DMS can be further divided into two subsystems: Physical System State-based DMS (P-DMS) and Environment State-based DMS (E-DMS) [1].

- P-DMS is the subsystem that computes the decisions based on the input from the physical system; for example, the control decisions that are made based on the sensor data from the machines.
- E-DMS is the subsystem that computes the decisions based on the inputs from the external environment. For example, the decisions that are made based on the market conditions, and unplanned stoppage of production due to unavailability of transportation of products.

The P-DMS is further partitioned into two subsystems: iP-DMS and oP-DMS:

- The iP-DMS includes all the P-DMS decision processes that can be modeled as a decision-free Petri net (iP stands for inside Petri net).
- The oP-DMS includes all the P-DMS decision processes that cannot be modeled as a decision-free Petri net (oP stands for outside Petri net).

Since we are primarily interested in modeling manufacturing system with Petri nets, we group the set of decision processes belonging E-DMS and oP-DMS into the subsystem called Outside-DMS (O-DMS). Thus, O-DMS = (E-DMS) U (oP-DMS) [1].

For practical modeling of a manufacturing system, not all the subsystems that are shown in the Fig. 1 are needed. The most needed and often used subsystems are the active and passive physical systems, and the iP-DMS and the O-DMS.

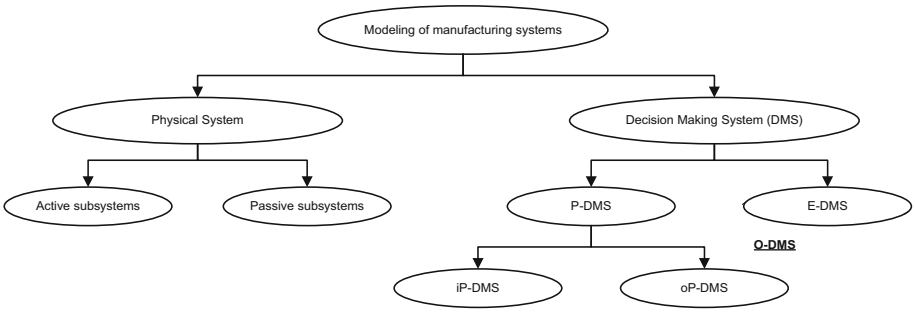

Fig. 1. The subsystems of a manufacturing system

2.3 Interaction Between O-DMS and iP-DMS

A Petri net model of a large manufacturing system consists of several modules that represent functional entities of the manufacturing systems. These Petri net modules makeup the iP-DMS; O-DMS is the decisions that are made outside the Petri net modules. The interaction between the iP-DMS and O-DMS subsystems happens through the control places [4], as shown in the Fig. 2.

In the Fig. 2, the Petri net model consists of two subsystems, the iP-DMS and the O-DMS. The iP-DMS consists of six Petri net modules, representing the arrival of raw materials, two machines, one robot for handling, a testing station, and a unit for transportation of products out of the manufacturing facility. The Petri net modules are controlled (enabled) by the control places (represented by two concentric circles), which are under the complete control of the O-DMS. Unlike ordinary places in a Petri net, the control places do not loose tokens when the input transitions (that are connected to the control places) fire. The tokens in the control places are only for enabling the respective transitions. The tokens are deposited into and removed from the control places by the O-DMS.

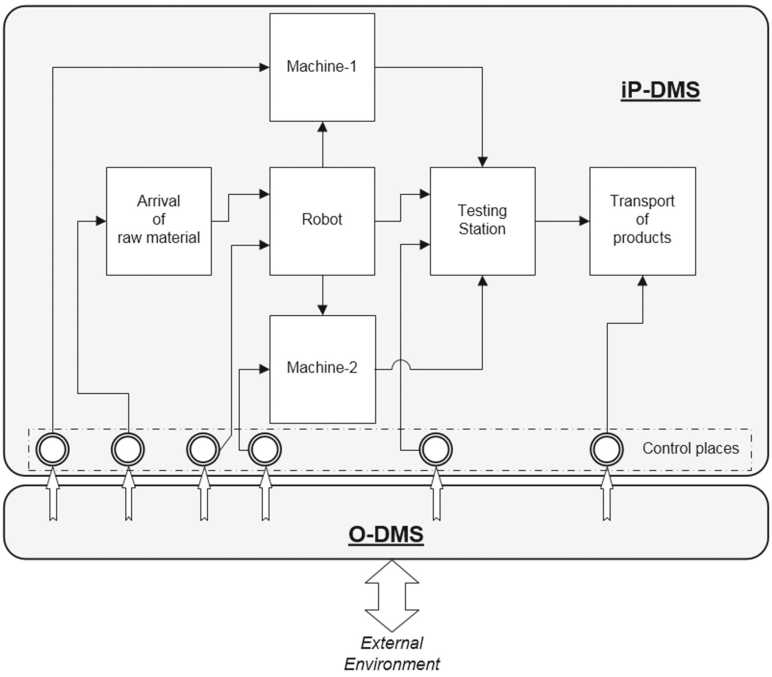

Fig. 2. Classical approach: the interaction between the iP-DMS and O-DMS subsystems happens through the control places

3 The Classical Approaches for Developing Petri Net Models of Manufacturing Systems

We start this section with an example to illustrate the classical approach for modeling a manufacturing system with the subsystems O-DMS and iP-DMS. Then, we show how the new approach can provide more expressive and yet compact Petri net models.

3.1 Modeling by the Classical Approach: An Example

Figure 3 shows the Petri net model of a simple Flexible Manufacturing System (FMS). This FMS consists of an input conveyor belt, an output conveyor belt, a horizontal CNC machine (H-CNC), a vertical CNC machine (V-CNC), and a testing station. The FMS also has a robot to move raw material and parts between the elements mentioned above. Since the model is so simple, the transitions and places are not grouped into modules as shown in the Fig. 3.

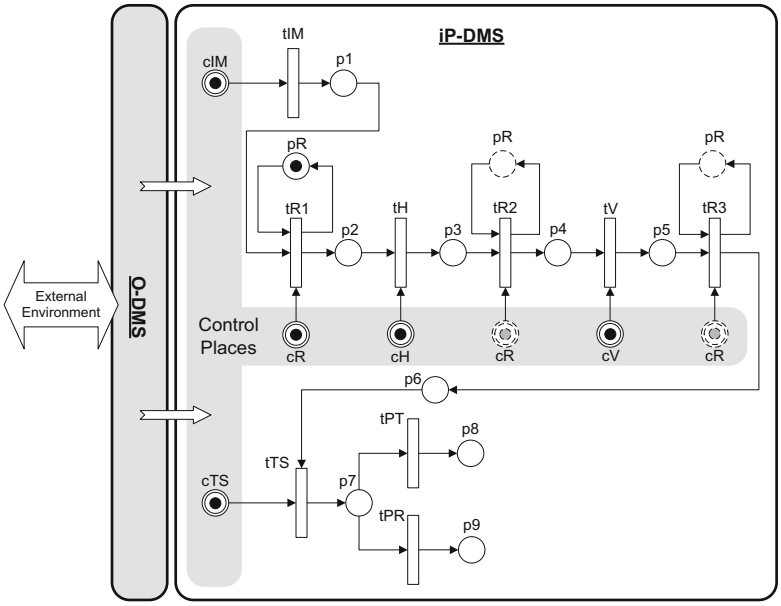

Fig. 3. The model of a simple Flexible Manufacturing System.

When the input material arrives (represented by the transition tIM), the robot picks them from the conveyor belt (tR1) and places into the H-CNC machine. The FMS performs a simple job with two operations: the first operation (tH) is performed by the H-CNC machine and the second one (tV) is performed by the V-CNC machine. After these two operations, the robot put the product into the testing station (tR3). If the product passes the quality test (tTS), it goes into (tPT) the output conveyor belt; otherwise, it will be discarded automatically (tPR).

Table 1. The elements of the Petri net model.

Part	Purpose	Part	Purpose
tIM	The flow of input raw materials	tPT	Transportation of quality products
tR1	Moving raw material from input conveyor belt into H-CNC	cIM	Control place for controlling the flow of input material
tH	Operation on the H-CNC	cR	Control place for enabling the robot
tR2	Moving parts from H-CNC into V-CNC	cH	Control place for enabling H-CNC
tV	Operation on the V-CNC	cV	Control place for enabling V-CNC
tR3	Moving parts from V-CNC into testing station	cTS	Control place for enabling the testing station
tTS	Tests on the testing station	p1-p9	Buffering places
tPR	Removal of disqualified products	-	-

Table 1 lists the transitions and the places involved in the Petri net model. In the Petri net model, the control place cR and the place pR are shown three times, in order to avoid cluttering of connections.

3.2 The Control Places

In the classical approaches, control places are the main mechanism for interfacing iP-DMS and O-DMS, as shown in the Figs. 2 and 3 [1, 3, 4]. A control place contains at most one token, where the token is deposited into and removed from by O-DMS. However, the implementation of the token addition and removal is not defined in any works. Also, the control places can only offer Boolean ON/OFF control, which is very limited in scope when it comes to modeling present day intelligent, complicated, and large-scale manufacturing systems.

4 A New Approach

In the previous section, the control place based mechanism was introduced for Petri nets based modeling of manufacturing systems. This mechanism has the following limitations:

1. The control places have to be treated as special places, as they do not behave like normal places. Thus, the mathematical tools that are so useful for analysis of Petri net models (such as reachability tree) become useless.
2. As stated earlier, for large Petri net model, the O-DMS has to control a large number of control places.
3. Most importantly, the control place based mechanism offers a very primitive control power, namely Boolean ON/OFF type control.

In the new approach proposed in this section, the control places are eliminated altogether. Instead, the transitions are controlled by the O-DMS. To allow O-DMS directly control the transitions inside the iP-DMS, each transition is equipped with pre- and post-processors. The new approach is known as the Activity-Oriented Petri Nets (AOPN) [5]. The software tool General purpose Petri net simulator (GPenSIM) is a realization of AOPN on the MATLAB platform; GPenSIM is freely available from the website [6]. Both the AOPN methodology and the software GPenSIM are developed by the author of this paper [2, 7].

4.1 Activity-Oriented Petri Nets (AOPN)

In the AOPN approach:

1. There are no control places. A Petri net model or module only consists of ordinary places, transitions, and arcs.
2. Every transition in the Petri net module can have a pre-processor. The pre-processor of a transition will be executed whenever the transition becomes enabled. Only if

the conditions coded in the pre-processor are satisfied, then the transition is allowed to start firing.
3. Every transition in the Petri net module can have a post-processor. The post-processor of a transition will be run immediately after the transition completes firing.
4. The O-DMS control every transition through the pre-processors and post-processors.
5. The transitions in a Petri net model usually represent active physical systems only (e.g. CNC machines). The passive physical systems (e.g. AGVs transporting materials), referred to as 'resources' in AOPN terminology, are not usually represented by transitions in the Petri net model.
6. When an active system (represented by a transition) needs a resource to perform an operation, the active system requests the O-DMS for the resource. If the resource is available, the O-DMS allocates the required resource to the transition so that the transition can start firing. When the transition finishes firing, it releases the resource back to O-DMS.
7. O-DMS also computes details about the resource usage: how many resources were used, their efficiency of utilization (line efficiency), the costs involved in the resource usage, etc.

The pre-processor and the post-processor are MATLAB files (M-files). Since these are M-files, we can write any code, as much as we want, for bidirectional information flow and control flow between O-DMS and the Petri net model (iP-DMS) (Fig. 4).

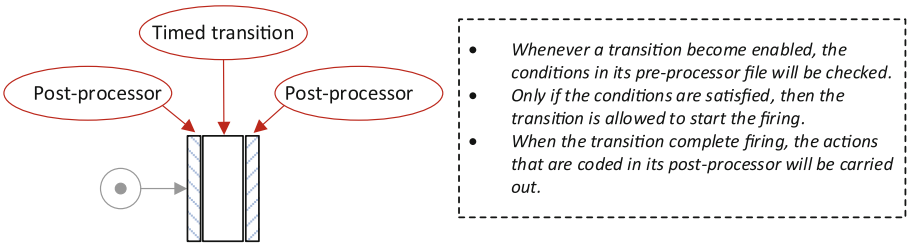

Fig. 4. Firing of a transition: the pre- and post-processor files.

Modeling a discrete event system by the AOPN approach consists of two phases [5]:

- Phase-I: Developing the static Petri net model
- Phase-II: Developing the run-time model

4.2 Phase-I: Developing the Static Petri Net Model

In the Phase-I, the static Petri Net model is developed. In this phase, mainly the activities are considered and they are represented by transitions interleaved with buffering places (e.g. places p10–p15 in the Fig. 5). The resources (passive physical systems) are grouped into two groups such as 'focal' resources and 'utility' resources. Only the focal resources are included in the static Petri Net model; the utility resources will be considered later in the phase-II (the run-time model). Thus, a compact Petri Net model is obtained with only the transitions representing the activities and, if there are any focal resources, they

will be represented by places. Using the tool GPenSIM, coding the static Petri net in phase-I will result in the Petri net definition file (PDF).

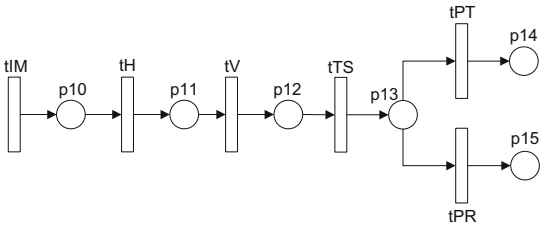

Fig. 5. The static Petri net model resulting after phase-I of the AOPN modeling approach.

Figure 5 shows the static Petri net model. In this model, the robot is assumed as a utility resource thus not shown in the model. The active physical systems such as H-CNC, V-CNC, and the testing station are represented by the transitions tH, tV, and tTS, respectively. Also, the arrival of raw materials is represented by tIM and the outflow of the testing station, namely accepted and rejected parts are represented by tPT and tPR, respectively.

The Petri net definition file (PDF) is the GPenSIM code for implementing (programming) the static Petri net model in GPenSIM software. The PDF for the static Petri net shown in the Fig. 5 is given in the Fig. 6.

```
% Conference ISPEM-Wroclaw, Sep 2017
% PDF: (fms_pdf)

function [png] = fms_pdf()

png.PN_name = 'A Simple FMS';

png.set_of_Ps = {'p10','p11','p12','p13','p14','p15'}; % set of places
png.set_of_Ts = {'tIM','tH','tV','tTS','tPT','tPR'}; % set of trans
png.set_of_As = {'tIM','p10',1,'p10','tH',1,'tH','p11',1,... %tIM and tH
                 'p11','tV',1, 'tV','p12',1, ...          % tV
                 'p12','tTS',1, 'tTS','p13',1, ...        % tTS
                 'p13','tPT',1, 'tPT','p14',1, ...        % tPT
                 'p13','tPR',1, 'tPR','p15',1, ...        % tPR
```

Fig. 6. PDF for the static Petri net model shown in the Fig. 5

4.3 Phase-II: Developing the Run-Time Model

In the phase-II of the AOPN approach, the run-time details that are not considered in the phase-I are added to the Petri Net model; e.g. transitions (activities) requesting, using, and releasing of the utility resources are coded in the run-time model (Fig. 7). Using the tool GPenSIM, the run-time details in the phase-II will result in the pre-processor file COMMON_PRE.m and the post-processor file COMMON_POST.m.

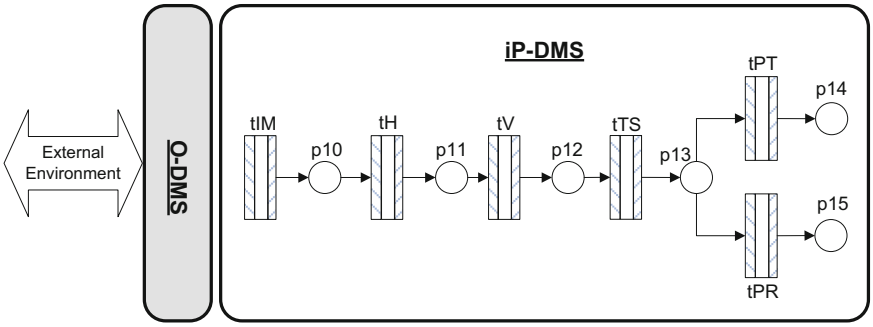

Fig. 7. The run-time Petri net model obtained after the Phase-II of the AOPN approach. The run-time details are coded in the M-files COMMON_PRE and COMMON_POST

In the COMMON_PRE file (Fig. 8), the following two conditions are coded:

- The transitions tH, tV, and tTS will be requesting the robot to load the part into the respective machine.
- About 2% of the products are rejected by the testing station.

```
function [fire, trans] = COMMON_PRE(trans)
% any conditions for enabled transitions are coded here

% if the transition is either tH, tV, or tTS
if ismember(trans.name, {'tH','tV','tS'}),
    % reuqest the robot to upload the part
    % fire, only if robot does its job
    fire = requestSR({'Robot',1});
    return
end

% if the transition is either tPT or tPR
if ismember(trans.name, {'tPT','tPR'}),
    % about 10% of products get rejected
    random_value = rand(); % random value between 0 and 1

    % if the transition is tPR
    if strcmp(trans.name, 'tPR'),
        fire = lt(random_value, 0.1); % tPR fires if value < 0.1
    else % this is 'tPT'
        fire = ge(random_value, 0.1); % tPT fires if value ≥ 0.1
    end
    return
end

fire = 1;  % allow all other transitions (e.g. 'tIM') to fire
```

Fig. 8. The COMMON_PRE file.

In the COMMON_POST file shown in the Fig. 9, the transitions tH, tV, and tTS will be releasing the robot after usage.

```
function [] = COMMON_POST(trans)
% any code for post-firing actions are coded here
% if the robot was aquired for loading the parts
% then release it after its service
release();
```

Fig. 9. The COMMON_POST file.

4.4 GPenSIM Code for Simulations

For simulation of the FMS problem with GPenSIM, four M-files are needed:

- Petri Net Definition File (PDF): this is the GPenSIM code of the static Petri net. For example, the PDF given in the Fig. 6 is the GPenSIM code for the static Petri net model shown in the Fig. 5.
- Main Simulation File (MSF): In this file, the initial dynamics (e.g. initial tokens in the places: none in this example), the firing times of transitions, and the availability of the resource 'Robot' are declared. The MSF is shown in the Fig. 10.

```
% Conference ISPEM-Wroclaw, Sep 2017
% Main Simulation File (MSF): fms.m
clear; clc; close all;
global global_info
global_info.STOP_AT = 350;

pns = pnstruct('fms_pdf'); % Declare the PDF

% Assign the initial dynamics
dyn.ft = {'tIM',10, 'allothers', 1}; % the firing times
dyn.re = {'Robot',1,inf};  % the systems resources

% combine the static graph with initial dynamics
pni = initialdynamics(pns, dyn);

% Run the simulations
sim = gpensim(pni);

% pront the results
plotp(sim, {'p14','p15'});
```

Fig. 10. The main simulation file.

- COMMON_PRE: In this file, the conditions for the enabled transitions to start firing are coded, mainly about reservation of the required resource 'Robot' by a transition, and using the resource if it is allocated.

- COMMON_POST: the post-firing actions of the transitions are coded here, usually releasing the resource 'Robot' after usage.

For reproducibility, the complete code for simulation (the four files mentioned above) is given on the web page [8]; interested readers are encouraged to visit the web page for downloading the codes and experimenting with them.

4.5 Simulation Results

Simulation results show that some products with the probability of 10% get rejected by the testing station. The rejected parts end up in the buffer p15, whereas the quality products are placed in the buffer p14 (Fig. 11).

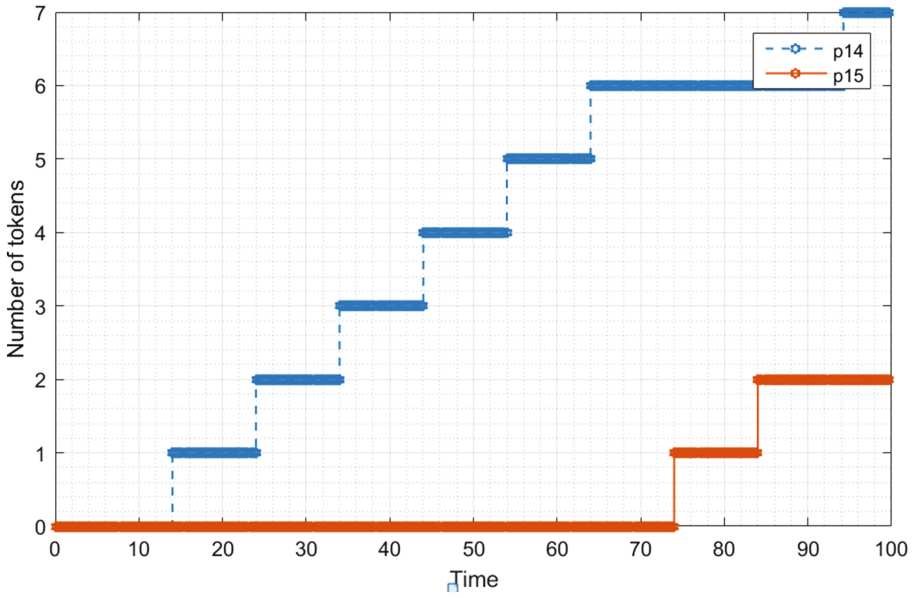

Fig. 11. Simulation results.

5 Discussion

The FMS example shows that the AOPN approach provides a simple two-way information and control flow between the Petri net model (iP-DMS) and the surrounding decision-making system (O-DMS) via the pre- and post-processors. The classical approaches that use control places for interfacing the iP-DMS and O-DMS provide only primitive Boolean ON/OFF control, whereas with the AOPN approach any control algorithm can be coded into the pre-processors and post-processors. The FMS example also shows that implementation of the Petri net models using GPenSIM usually results

in compact code as none of the files (PDF, MSF, and the pre- and post-processors) is more than half page long.

In summary, the duality – the compact Petri net models because of the activity-oriented view and the powerful and flexible resource management by the O-DMS, enables modeling, simulation, performance analysis, and control of any discrete event system.

6 Conclusion

GPenSIM is the software realization of the new AOPN approach. Though AOPN approach and the software GPenSIM are new, there are Doctoral dissertations, and many Master theses are done based on AOPN/GPenSIM. GPenSIM is used to solve many industrial and research problems in diverse fields of engineering. For example, in discrete manufacturing systems and production planning, airport capacity modeling, fish supply chain, Speech recognition, computer gaming, Marine-Diesel Engine Control, Service-Oriented Architecture, Grid-Computing, E-commerce, and Bluetooth Devices, to name a few [9, 10].

Acknowledgement. This paper was written when the author was staying at the Silesian University of Technology, Poland, for his sabbatical leave. The author wants to thank the Institute of Engineering Processes Automation and Integrated Manufacturing Systems of the Faculty of Mechanical Engineering, for hosting him.

References

1. DiCesare, F., Harhalakis, G., Proth, J.M., Silva, M., Vernadat, F.B.: Practice of Petri nets in manufacturing, p. 8. Chapman & Hall, London (1993)
2. Davidrajuh, R.: Activity-Oriented Petri Net for scheduling of resources. In: IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1201–1206. IEEE (2012)
3. Peterson, J.L.: Petri nets. ACM Comput. Surv. (CSUR) **9**(3), 223–252 (1977)
4. Holloway, L.E., Krogh, B.H.: Synthesis of feedback control logic for a class of controlled Petri nets. IEEE Trans. Autom. Control **35**(5), 514–523 (1990)
5. Davidrajuh, R.: Modeling resource management problems with activity-Oriented Petri Nets. In: Sixth UKSim/AMSS European Symposium on Computer Modeling and Simulation (EMS), pp. 179–184. IEEE (2012)
6. GPenSIM User Manual. http://www.davidrajuh/gpensim/
7. Davidrajuh, R.: Developing a new Petri net tool for simulation of discrete event systems. In: 2008 Second Asia International Conference on Modeling & Simulation, AICMS 2008, pp. 861–866. IEEE, May 2008
8. Simulation code. http://www.davidrajuh.net/gpensim/Conf/2017-ISPEM/
9. Skolud, B., Krenczyk, D., Davidrajuh, R.: Solving repetitive production planning problems. An approach based on Activity-Oriented Petri Nets. In: International Conference on EUropean Transnational Education, pp. 397–407. Springer International Publishing, October 2016
10. Davidrajuh, R., Lin, B.: Exploring airport traffic capability using Petri net based model. Expert Syst. Appl. **38**(9), 10923–10931 (2011)