ICIC Express Letters
Part B: Applications
Volume **9**, Number **10**, October **2018**

ICIC International ©2018 ISSN 2185-2766
pp. **1033–1040**

PETRI NETS SIMULATION FOR OPERATIONAL ANALYTICS BASED ON PROCESS DATA: AN OVERVIEW

Natanael Yabes Wirawan[1], Riska Asriana Sutrisnowati[2], Sunghyun Sim[3]
Nur Ichsan Utama[3], Nur Ahmad Wahid[1], Taufik Nur Adi[3], Yulim Choi[3]
Imam Mustafa Kamal[1], Iq Reviessay Pulshashi[3] and Hyerim Bae[3,*]

[1]Department of Big Data
[3]Department of Industrial Engineering
Pusan National University
30-san Jangjeon-dong, Geumjeong-gu, Busan 46241, Korea
{ yabes.wirawan; ssh; nichsan; nawa410; taufiknuradi; flamethrower }@pusan.ac.kr
{ imamkamal; pulshashi }@pusan.ac.kr; *Corresponding author: hrbae@pusan.ac.kr

[2]Dong-Nam Grand ICT R&D Center
Pusan National University
QB e-Centum, 90 Centum Jungang-ro Haendae-gu, Busan 48059, Korea
riska@pusan.ac.kr

Received March 2018; accepted June 2018

Abstract. *This paper highlights an initial study on a framework for developing a semi-automatic Petri nets simulation model for operational analytics. Starting from process/event data gathering from companies' software information systems, we calculate the simulation parameters (e.g., distribution parameters, and simulation logic). Afterwards, we build the model by integrating the simulation parameters and process execution logics. Also, we incorporate into the framework several algorithms that are designed to handle large-sized data.*
Keywords: Process mining, Simulation, Petri nets, Event logs, Distributed processing

1. **Introduction.** In highly complex processes, researchers and managers use simulation methods to represent several scenarios that seem impractical or expensive in the real world. This involves understanding of the underlying process, gathering of data, interviewing of the domain experts in the relevant fields and observing how certain rules are applied to the process. These methods sometimes fail when the data is incomplete or the real problem is so complex that simplification sacrifices correctness and performances. To say the least, developing a simulation model is not an easy task [8].

Researchers [6,7] have used Petri nets as tools for discrete event simulations. Petri nets can represent both the static and dynamic aspects of a process. The process itself represents the static aspect, while the consuming and producing tokens represent the dynamic aspect. Originally, researchers used Petri nets to model concurrency, communication and synchronization [5]. One famous tool called Colored Petri Nets Tools (CPNTools) [10] provides a canvas for creating, editing, simulating and reporting a simulation model. However, CPNTools did not provide parallel processing or distributed computing capability to handle large data in distributed manner. Still, it is only available as a desktop application which makes it difficult to be scaled and maintained.

Thanks to increased data sizes and improvements in computing power over the past decade, simulation techniques have been processing more and more data. Prior to the use of big data, the fields of biology, chemistry, and aerospace engineering had already harnessed GPGPU (General-Purpose computing on Graphics Processing Unit)-based parallel

DOI: 10.24507/icicelb.09.10.1033

processing for visualization and numerical analysis. Recently in the Petri nets simulation, researchers have been taking advantage of improved computing power available with parallelize Petri nets [2], hierarchical-timed colored Petri nets [7], and others. However, there is no Petri nets simulation that can efficiently extract knowledge from operational big data. To meet this challenge, this paper introduces a framework for building of a simulation model semi-automatically from event logs by taking account of data size.

This paper is organized as follows. Section 2 presents the related work and Section 3 discusses the proposed simulation framework. Section 4 draws conclusions.

2. Related Work.

2.1. Simulation.

Computer simulation is a powerful tool with which to observe and validate what is going on in real-life processes. By simulating a real process, we are able to see what can be enhanced and improved under any circumstances [13]. Simulation has been implemented in various disciplines, from military science to healthcare, gaming and industry. Nowadays, there are many popular simulation tools, including SIMULA, ARENA, Simul8, COSA, Aris, and others. However, such tools are complex and expensive, requiring not only large investments for their purchase but also experts to operate them.

In order to run a simulation, we need a good model (traditionally created by an expert) that represents the real world. And as a company needs to continually increase its revenue, it also needs to increase its throughput every year. This means that it should change its processes annually and correspondingly, and models need to be changed year by year. To solve this problem, Rozinat et al. [11] proposed semi-automatically generation of a simulation model from event logs. Specifically, they used some process mining techniques such as control-flow discovery, role discovery, decision point analysis and performance analysis to generate Petri nets model. Afterwards, they simulate it by using CPNTools [10] in order to evaluate its performance. In addition, the tool enables us to predict the Key Performance Indicators (KPI) from different model scenarios that users create.

In business process simulation, there are two types of simulations, namely those for long-term and short-term planning, respectively. Long-term simulation is used to simulate the long-term effects of certain decisions, while short-term simulation is commonly used to assist decision making in current situations. Based on the types of models, there are discrete and continuous simulations. In discrete-event simulation, the states of variables change only at a countable number of points in time (integers), whereas in continuous simulation, the states of variables change continuously (an infinite number of states). This research is dedicated to discrete event simulation for the Petri nets model.

2.2. Petri nets and colored Petri nets.

In the area of process mining research, Petri nets is a common standard in which processes are modeled. Petri nets was invented in 1962 by Carl Adam Petri for the purpose of modeling discrete distributed systems. A simple graphical representation of the progress of the process allows one to see at a glance, many of the synchronous events that occur during the sequential part of a process and to capture and identify the structural problems of that process. A Petri net consists of four components: place, transition, arc and token. Figure 1 shows an example of a process model using Petri nets. In the figure, the circles, rectangles, and arrows indicate the place, transition, and arc, respectively. Also, the point in the circle is the token, the dynamic aspect of the model.

A place represents a possible "state" within the system, a transition is an "event" that triggers a state change, an arc represents the flow between a place and a transition, and a token can be considered as an entity within the system. A transition generally refers to one event. In order for this event to occur, one or more conditions must be met. A place has information about the condition(s). An arc represents the relationship

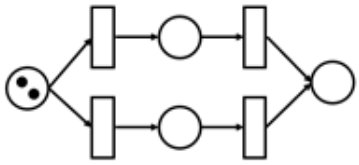

FIGURE 1. Example of Petri nets

between a transition and a place. There are two types of arc, which differ according to the connection direction: the input arc connects a transition to a place and the output arc connects a place to a transition. There is a source transition and a sink transition in a special transition. A source transition is a transition without an input place, and a sink transition is a transition without an output place. To construct a Petri net, there must be finite number of transitions and places, and they must be connected to an arc.

Due to the general nature of Petri nets, it can be applied to a variety of applications. However, when applied to real-world problems, the Petri net paradigm of places, transitions, arcs, and tokens cannot intuitively be understood by users [3]. To overcome this limitation, Colored Petri Nets (CPN) was proposed by Kurt Jensen in 1980 [4]. CPN is an extension of ordinary Petri nets. In CPN, colors are associated with tokens, and the transition will fire according to a set of rules that match the appropriate colors. Jensen [4] has introduced and defined the CPN, the main idea of which is a relation between an occurrence color and a token color. The relation is defined by functions attached to the arcs. Also, the CPN attaches a set of possible token colors to each place and a set of possible occurrence colors for each transition.

2.3. **Petri nets simulation tools.** The tools that can be used for editing, simulating and analyzing CPN are PetriDotNet [14] and CPNTools [10]. However, up until this study is conducted, we cannot study PetriDotNet source code since it is not an open source program. Thus, for the remaining of the paper, we exclusively use and extend CPNTools. CPNTools has its own workspace management which is a workspace that occupies the whole screen and contains a window-like object called binders. This workspace is intended to provide easy access for management of a large number of pages that are typically found in industrial-sized CPN. Binders contain sheets, each of which is equivalent to a window in a traditional environment. A sheet provides a view of either a page from a CPN, declarations, or a set of tools. Each sheet has a tab similar to those found in tabbed dialogs. The CPN Tools has some features [10] to support several user interactions for manipulation of objects in binders, such as direct manipulation, bi-manual manipulation, marking menus, keyboard input, pallets, tool glasses, the index, and magnetic guidelines.

CPNTools supports two types of simulations, which are interactive and automatic. In the interactive simulation, the user is in full control and determines the individual steps by selecting enabled transitions in the current state. In automatic simulation, the user specifies the number of steps that are to be executed and/or sets a number of stop criteria and breakpoints. The simulator then automatically executes the model without user interaction by making random selections of enabled transitions in the states encountered.

In the perspective of usability, CPNTools has many things requiring improvement for enhanced ease of learning and utilization. For the use of a specific domain, CPNTools still requires a third-party software (e.g., CPNTools Export [11] in ProM [9] in the process mining domain, though it requires manual integration of some plugins).

CPNTools is a desktop-based program. Desktop-based programs have limitations in that they have to be developed and installed on a particular operating system and usually have some specific program and hardware requirements needed to make them work properly. Also, when there is a specific update, it has to be manually installed to the

client computer and may require hardware updates or other changes in order to work, which certainly is not very convenient. All of these problems can be resolved, however, if the program is made with web-based architecture. Additionally, the use of existing web browsers and their multimedia capabilities has allowed for creation of more interactive, media-rich user interfaces.

3. Simulation Framework.

3.1. The architecture.

The architecture of our simulation framework is presented in Figure 2. The core architecture consists of four main parts, namely the model integration, simulator engine, system interface (REST API), and UI engine. The simulation framework leverages parallel processing and distributed computing by using Distributed File System (DFS) technology to store all data and information (e.g., the event logs, process model, simulation configuration, and simulation model) needed to simulate business processes.

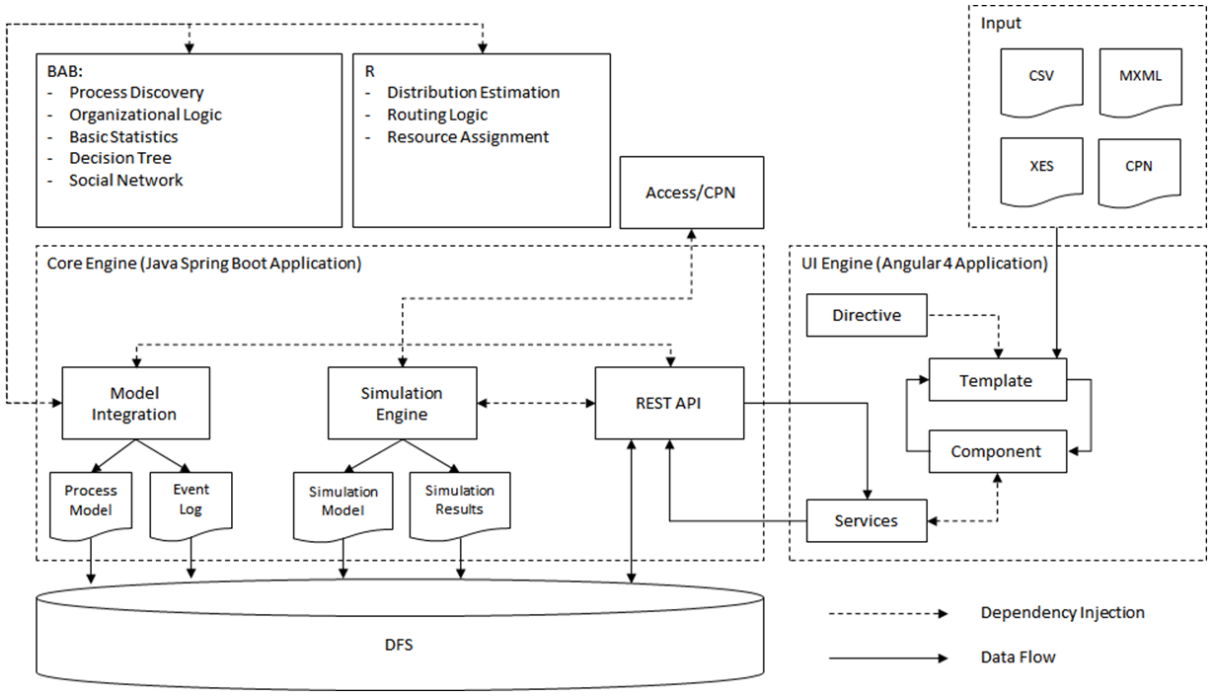

FIGURE 2. Architecture of Petri nets simulation framework

The model integration is a binding to the BAB framework [12] and R programming language. The BAB framework is a process mining tool created in our previous work to enable users to analyze their data using process mining regardless of the platform [12]. It has a subset of process mining features such as process discovery, organizational logic, basic statistics, decision tree, and social network analysis. Meanwhile, the R programming language will give additional advantages in terms of distribution estimation calculation for each activity/equipment discovered in the model, routing logic rules, and resource assignment rules.

The simulator engine is a binding to the CPNTools application that performs the token replay simulation. Given a .cpn file simulation model, the engine will call up a routine in Access/CPN (i.e., CPNTools from the Java library) to communicate with the CPNTools core engine, namely Standard ML, in order to perform the simulation [15]. The results of token-replay will be available for the REST API in text file format and are stored in the DFS.

3.2. **Simulation model integration.** In BAB [12], there is an already-implemented distributed version of the algorithms needed to build the initial model (i.e., process discovery algorithms, organizational logic, basic statistics and decision logic). To complete the simulation model, we need to incorporate more information such as the distribution estimator, routing logic and resource assignment rules using R programming languages. Once all information is gathered, we integrate each partial analysis into one model in the .cpn format. Since the extension analysis of BAB is already described in [12], in this section we will describe in detail the add-ons built using the R programming languages.

First, a distribution estimator tries to estimate the processing time of each job/task that is later incorporated into the transitions of a Petri net model. To predict it, we estimate the processing time distribution using the distribution-fitting method and the Bayesian updating method.

We can apply two methods for fitting the distribution depending on the properties of the distribution, which are pure probability and mixed probability. For fitting a pure probability distribution, we use the Kolmogorov Smirnov Test (KS-Test), which is one of the fitness tests to find the appropriate distribution from the data. We use the discrete probability distributions (Bernoulli, Binomial, Uniform, Geometric, Hypergeometric, Negative binomial, Poisson) and continuous probability distributions (Normal, Log-normal, Beta, Cauchy, Chi-square, Exponential, Gamma, Pareto, F, T) for fitting between the data and the appropriate distribution. For fitting a mixed-probability distribution, we use EM algorithms to fit multiple mixed normal distributions or mixed log-normal distributions. Since there are many real-world cases of mixed distribution rather than of pure data, there would be many errors if we use a pure probability distribution when the data has a mixed distribution.

In the initial stage of simulation, the working time distribution of equipment, which generally is estimated from data, cannot be accurately known. Even if we find the working time distribution of each equipment from the extracted event logs, it cannot be seen as an exact distribution of the jobs that take place in the following time. However, if the simulation is in the stabilized state, we can estimate each equipment's future processing time distribution accurately. This is accomplished using a Bayesian learning and updating method, as illustrated in Figure 3.

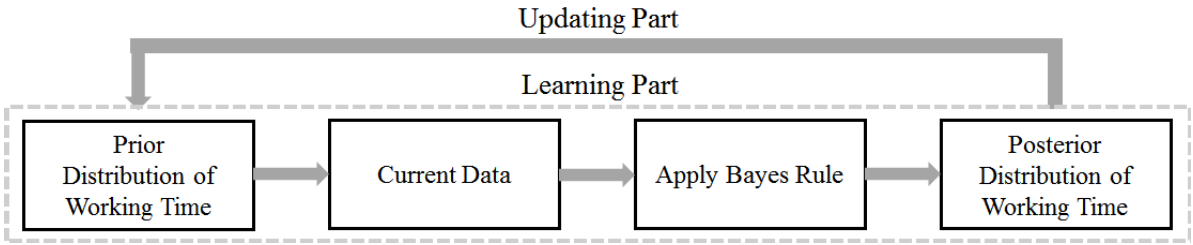

FIGURE 3. Bayesian learning and updating method

The second add-on built using the R programming languages is resource assignment. With the aid of organizational logic and social networks we calculate the rules on how a resource is selected during the simulation, using the similarity distance and probability among resources in a certain job/task/activity.

Lastly, there is a routing logic add-on. From Petri nets point of view, the routing logic corresponds to the determination where a token should be fired when an arc is divided into several places. The method for determining the routing logic is as follows. First, we need to identify the part of the process model where the process breaks into different points. Afterwards, with the aid of the decision tree, we analyze the probability from the breaking point by using the probabilistic frequency and Bayesian probability for developing the routing rule.

Once all of the information is gathered, it will be integrated into a simulation model.

3.3. **Simulation engine.** The simulation engine in this application is worked by utilizing the CPNTools core engine via Access/CPN. Access/CPN has two interfaces that were built in java programming and Standard Meta-Language [16]. The current version of Access/CPN is Access/CPN 2, which was developed by modifying its first released Access/CPN 1 [15]. By using the Access/CPN 1 interface, our simulation engine can communicate with the generated model specific simulator code to perform a simulation by executing tokens in transitions and obtaining the resultant marking of places. By using the Access/CPN 2 interface, our simulation engine can perform co-simulation between the simulation model and Java programming languages' classes to monitor the situation that has changed in a specific place and transition during the simulation [16], and this is very useful when we want to monitor performance analysis in a specific node of the CPN model.

As already explained previously (in Section 3.1), our simulation engine will use .cpn file simulation model that is generated by simulation model integration part. The simulation model will be loaded into the CPNTools core engine by using Access/CPN, so that our simulation engine will be able to monitor the marking changes of places and situation in a specific node (places and transitions) and obtain the results of performance analysis. All of the results of the process in a simulation engine will be converted to the JSON format and can be accessed in REST API to make possible communication between the simulation engine and the UI.

3.4. **Sytem interface: REST API.** The system interface is a REST API, which binds to the model integration and simulation engine. The system interface will retrieve all necessary information related to the event logs, process model, simulation model, and simulation result. This information can be the metadata of the given files, the algorithm used to generate the process model, the simulation token replay result, or other analyses results. All of this information is transmitted to the UI engine in the JSON format as described in the previous section.

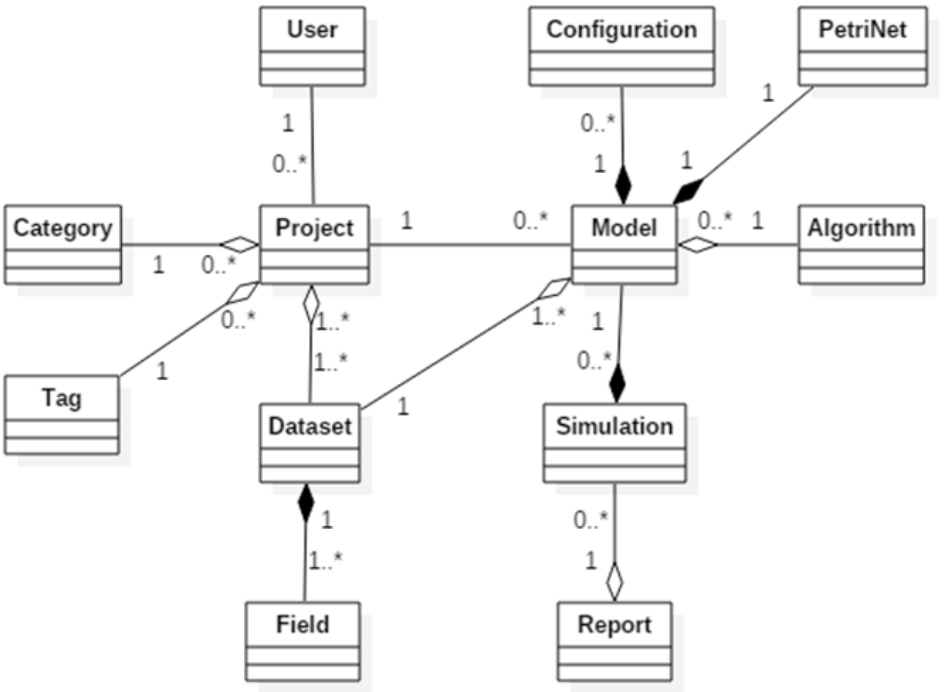

FIGURE 4. Domain model of simulation UI engine

In order to ensure that all of the information components are connected to each other, we propose a domain model to organize all information in the UI engine. We develop the UI engine using Angular 4 application [1] to obtain, display, and visualize all necessary information in a client browser, as shown in Figure 4. A project is created by a user. Each project has metadata information such as category and tags. Each project can also have more than one dataset, each corresponding to an event log. Datasets consist of many fields. A project can have zero or more than zero models. The model contains information such as simulation configuration (i.e., the fitting distribution, organizational miner, basic statistics, decision point, and KPI), the process discovery algorithm (what algorithm is used to generate the model, the parameters of the algorithm, etc.), and the Petri nets model information (the process model). A model may be used in different simulation scenarios, which is to say that one model can be used to generate many simulation models. The simulation model may have zero or more than zero reports, which consist of information, for example, KPI results.

4. **Conclusions.** This paper aims to facilitate simulation of operational data using Petri nets and to accommodate, thereby, the growing data from software information systems. As an initial step, a framework for model building and simulation is described in detail. The core architecture of the simulation framework consists of four main parts, namely the model integration, simulator engine, system interface (REST API), and UI engine. The model integration is a binding to the BAB framework (to generate process model from a given event log) and the R programming language (performing distribution estimation, routing logic, and resource assignment). The simulator engine is a binding to a CPNTools application, which performs the token replay simulation. The results of the token-replay will be available for the REST API in a text file format and be stored in the DFS. The REST API will retrieve all necessary information related to the event logs, process model, simulation model, and simulation result. The UI engine will display all necessary information related to simulation activities and will additionally display all information from the event logs, process model, simulation model, simulation token replay, and simulation report. These achievements notwithstanding, we still face challenges in dealing with the scaling of the simulation engine, the partitioning of a response file such that the browser does not freeze during simulation or heavy loading of data, and, eventually, the reduction of network time during file transfer to/from the client-side. These challenges will be taken into consideration in future works.

Acknowledgment. This research was partly supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2017-2016-0-00318) supervised by the IITP (Institute for Information & Communications Technology Promotion) and National Research Foundation of Korea (NRF) grant funded by the Korean government (MEST) (No. NRF-2015R1D1A1A090613 31).

REFERENCES

[1] Angular, *Angular Docs*, https://angular.io/, 2018.
[2] G. Chalkidis, M. Nagasaki and S. Miyano, High performance hybrid functional Petri net simulations of biological pathway models on CUDA, *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol.8, pp.1545-1556, 2011.
[3] K. Jensen, High-level Petri nets, in *Applications and Theory of Petri Nets*, Springer, 1983.
[4] K. Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, Springer Science & Business Media, vol.1, 2013.
[5] K. Jensen and L. M. Kristensen, *Coloured Petri Nets Modelling and Validation of Concurrent Systems*, Springer-Verlag Berlin Heidelberg, 2009.

[6] M. Kamath and N. Viswanadham, Applications of Petri net based models in the modeling and analysis of flexible manufacturing systems, *The 1986 International Conference of Robotics and Automation*, IEEE Computer Society Press, vol.1, pp.312-317, 1986.
[7] Y. Q. Lv, C. K. M. Lee, Z. Wu, H. K. Chan and W. H. Ip, Priority-based distributed manufacturing process modeling via hierarchical timed color Petri net, *IEEE Trans. Industrial Informatics*, vol.9, pp.1836-1846, 2013.
[8] S. Park, H. Kim, D. Kang and D. H. Bae, Developing a simulation model using a SPEM-based process model and analytical models, *Lecture Notes in Business Information Processing*, vol.10, 2008.
[9] Process Mining Group Math & CS Department, *Process Mining Research Tools Application*, http://www.processmining.org/prom/start, 2009.
[10] A. V. Ratzer, L. Wells, H. M. Lassen, M. Laursen, J. F. Qvortrup, M. S. Stissing, M. Westergaard, S. Christensen and K. Jensen, CPN tools for editing, simulating and analysing coloured Petri nets, *The 24th International Conference on Applications and Theory of Petri Nets (Petri Nets 2003)*, vol.2679, pp.450-462, 2003.
[11] A. Rozinat, R. S. Mans, M. Song and W. M. P. Van Der Aalst, Discovering simulation models, *Information Systems*, vol.34, pp.305-327, 2009.
[12] R. A. Sutrisnowati, H. Bae, I. R. Pulshashi, A. D. A. Putra, W. A. Prastyabudi, S. Park, B. Joo and Y. Choi, BAB framework: Process mining on cloud, *Procedia Computer Science*, vol.72, pp.453-460, 2015.
[13] W. M. P. Van Der Aalst, Business process simulation revisited, *Lecture Notes in Business Information Processing*, vol.63, 2010.
[14] A. Vörös, D. Darvas, Á. Hajdu, A. Klenik, K. Marussy, V. Molnár, T. Bartha and I. Majzik, Industrial applications of the PetriDotNet modelling and analysis tool, *Science of Computer Programming*, 2017.
[15] M. Westergaard, Access/CPN 2.0: A high-level interface to coloured Petri nets models, *The 32nd International Conference on Applications and Theory of Petri Nets (Petri Nets 2011)*, vol.6709, pp.328-337, 2011.
[16] M. Westergaard and L. M. Kristensen, The access/CPN framework: A tool for interacting with the CPN tools simulator, *The 30th International Conference on Applications and Theory of Petri Nets (Petri Nets 2009)*, vol.5606, pp.313-322, 2009.