ICIC Express Letters
Part B: Applications
Volume **13**, Number **7**, July **2022**

ICIC International ©2022 ISSN 2185-2766
pp. **725–733**

COLORED PETRI NETS SIMULATION ON THE CLOUD WITH PYTHON: A FRAMEWORK OVERVIEW

Nur Ichsan Utama[1], Imam Mustafa Kamal[2], Alif Nur Iman[3]
Dohee Kim[1], Hyerim Bae[3,*] and Changwoo Hong[4]

[1]Department of Industrial Engineering
[2]Department of Big Data
[3]Industrial Data Science and Engineering, Department of Industrial Engineering
Pusan National University
2, Busandaehak-ro 63beon-gil, Geumjeong-gu, Busan 46241, Korea
{ nichsan; imamkamal; alifnuriman; kimdohee }@pusan.ac.kr
*Corresponding author: hrbae@pusan.ac.kr

[4]Cyber Logitec 396
Worldcup-bukro
Mapo-gu, Seoul 03925, Korea
cwhong@cyberlogitec.com

Received November 2021; accepted January 2022

Abstract. *This paper presents an initial study and implementation of a simulation framework based on Colored Petri Nets model. We implement this study based on cloud applications that use python programming language mainly for the backend services and React-based framework for its frontend application. The implementation incorporates a process mining technique to construct the flow of the simulation model as a backbone from event logs. The part related to resource assignment is constructed based on social network algorithm also in the process mining field. Other parts of the simulation model, such as performance analysis, resource allocation, decision point of activities, and arrival cases distribution, can be added flexibly using two mechanisms, hardcoded or plug-unplug from pickle file. Based on this mechanism, it is possible to use the result of an instance trained machine learning model that is trained outside of the application as long as it complies with the input-output criteria defined by our application.*
Keywords: Process mining, Simulation, Colored Petri Nets, Machine learning

1. **Introduction.** In the current manufacturing environment, almost all activity of process is feasible to be recorded. The data recorded later can be analyzed in realtime or offline for further analysis by the manufacturing, depending on the requirement. One of the possible uses of these data is to create a data-driven simulation model. Previously, the simulation model was derived based on the data usually constructed by the expert. Nowadays, software such as [1,2] is able to create simulation models automatically from the log data of the process without much human expert intervention. The type of simulation model generated based on that kind of data is Discrete-Event Simulation (DES). In DES, an event is typically defined as a specific change in the state of the system at a particular time [3]. The log data of the process usually also have information about an event that make this data useful for constructing the DES model. DES is not the only type of simulation, but it is one of the most popular existing simulation types used by many researchers and practitioners [3].

The log data of the process that records an event is called an event log. One of the data science fields that use this kind of data as the main input in its analysis is process mining. Process mining is an analytical discipline for discovering, monitoring, and improving the

DOI: 10.24507/icicelb.13.07.725

process. Process mining is both data-driven and process-centric [4]. In process mining, each event generally refers to an activity that occurs at a particular time for specific cases [5]. Process mining and DES, if we consider them from the event log perspective, are closely related. As stated before, the input for process mining is an event log [4]. Moreover, artificial event data generated from running DES also can be used as an input in process mining. Therefore, combining process mining and simulation is reasonable, such that the results of the two can complement each other as suggested [5].

The first practical approach that combined process mining and simulation based on the references that we got was proposed in 2009 [6]. The authors used a process model based on a Petri net representation discovered from an event log as a direction for token flow in the simulation process [7]. The authors used one of the process discovery algorithms in process mining, which is the alpha algorithm, to mine the workflow process from the event log [7]. The discovered process model represents a control-flow perspective, combined with other analysis techniques such as decision point analysis, organizational miner, and statistical analysis to generate the simulation model [6].

After the process model is discovered, sometimes the structure will contain some of the places in the Petri Nets that behave like an XOR split gate. In this case, we need to determine the split decision. The split decision can be based on frequency probability or data. If we consider using data, decision point analysis is used to mine decisions based on the attribute data that influence the choices made in the process based on the past process executions [8]. Organizational miner is conducted to mine resource assignments in a specific activity. Several algorithms, such as default mining, doing similar-task, and agglomerative hierarchical clustering, can be used to perform this resource assignment [9]. Statistical analysis is used to analyze the performance based on the event log. This analysis can also be used to fit the distribution of several parameters for simulation, such as the waiting time of activity, the execution time of activity and the arrival time of cases [6].

When we get an event log, we assume that the data inside that log is complete and represents what happened in the real operation. However, if a considerable data in the event log that we required to determine the parameters of the simulation is somehow not complete or missing, the missing data can be predicted using machine learning, such as done by the authors that first change the representation of the event log into an image [10]. Incorporating machine learning is not the only way to handle the missing data. The other researchers used a statistical method to handle this missing data by modifying the popular statistical algorithm Multivariate Imputation by Chained Equation (MiCE) [11]. Other research in process mining tried to incorporate semantic for annotation and modelling of domain process [12].

Although the idea of creating an automatic simulation model based on process mining is promising, many practical implementations do not consider modules to be embedded flexibly based on the cloud application. Many applications use pre-built analytic algorithms embedded in the app. In this current implementation, we created a modular module that can be added to the simulation model of our cloud-based application. Our study attempts to complement the previous study [13] by adding a modular module to our cloud application.

The main contribution of our research is as follows.

- We propose an automatic simulation model generator from event log data with a modular module to be plug-unplug in the cloud application.
- With the modular module, any python object, including an instance of a machine learning model, is possible to be loaded into our python cloud application.
- With this modular module, adding a new algorithm in the simulation model can be done loosely coupled by another user (collaborative).

The remainder of this paper is structured as follows. Section 2 discusses some terminology, provides an overview of the related work, and presents a running example. Section 3 outlines the simulation framework. Section 4 describes the explanation about the mechanism of how we run the simulation starting from the event logs. The result from the method in Section 4 is empirically evaluated in Section 5. The paper ends with a conclusion in Section 6.

2. Related Work.

In this section, we describe the preliminary of our study. It contains a definition and overview of process mining, event logs, process model, performance analysis, resource assignment and allocation, and Colored Petri Nets.

2.1. Process mining.

Process mining aims to discover, monitor and improve real processes by extracting knowledge from event logs [14]. Process mining has been used in the field of process management to support the analysis of business processes. During process mining, specific data mining algorithms are applied to event log data in order to identify trends, patterns, and details contained in event logs recorded by an information system. The starting point for process mining is an event log. Later, the output from the process mining algorithm that used event log as an input can be as an input for another process mining algorithm.

There are three main classes of process mining techniques, namely process discovery, conformance checking, and performance analysis. This classification is based on whether there is a prior model and, if so, how the prior model is used during process mining [4].

2.2. Event logs.

Based on previous information, we already know that an event log is an input for process mining. This section will define an event log, event properties, and traces. An event is described by some unique identifier and can have several properties. An event log is a set of events. Each event in the log is linked to a particular trace that represents a process instance and is globally unique, i.e., the same event cannot occur twice in a log. If in the trace we found two events with the same activity, there must be any other properties different between those two events, such as timestamp, and resource.

2.3. Process model.

Process model can be created manually or discovered from the event logs using several process discovery algorithm based on process mining techniques. In this part, we just explain one of the notations standards of the process model, which is Petri Nets. A Petri Net is a directed bipartite graph with two types of nodes called places and transitions. The places and transitions are connected to each other with directed arcs. Arcs connecting two nodes of the same type to each other are forbidden. So, an arc will be connecting only place-transition and vice versa. In a Petri Net, places are graphically represented by circles, and transitions are represented by rectangles. The elements of places and transitions are referred to as nodes. The representation of Petri Net is shown in Figure 1 below.

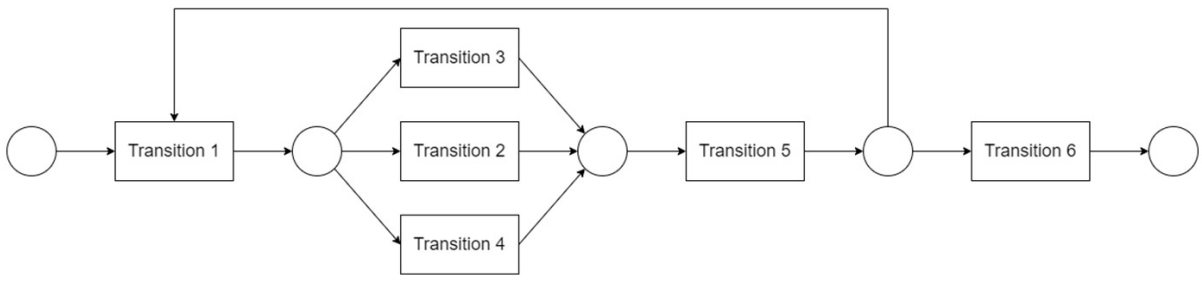

FIGURE 1. Example of Petri Net representation

2.4. **Performance analysis.** Performance analysis in process mining is to measure the efficiency and effectiveness of an activity or an entire process. It is related to the entire process, some performance that can be measured such as throughput, average cycle time, and average lead time. By analyzing this performance, we can measure productivity in some stage of the process related to a specific department, resource, or trace.

2.5. **Resource assignment and allocation.** Resource assignment is usually related to the assignment of resources to a specific task or activity. In comparison, resource allocation determines which resource should be allocated to execute a specific work item at a specific time. From the process mining perspective, this execution of work items can be related to the activity. In the process mining field, one of the algorithms that try to map between resource and activity is the organizational miner. The organizational miner is worked by clustering the resources into several organizational units and then assigning the organizational unit to specific activities.

2.6. **CPN.** Coloured Petri Nets (CPN) provide a modelling language that combines Petri Nets with the functional programming language [15]. Petri Nets model the basic behaviour of a process while the programming language handles the definition and manipulation of data values. In a CPN model, all the tokens are distinguishable by assigning each token a certain value, referred to as a colour. Figure 2 shows a fragment of a CPN model, which consists of two places and one transition. The circles in the model indicate the places in a Petri Net, and the boxes indicate the transitions. Each place in a Coloured Petri Net has a certain type. Each transition can have certain properties such as guard, action, and priority. Guard is represented as the conditional execution of the transition.

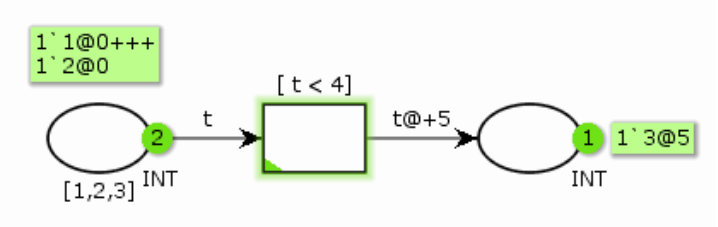

FIGURE 2. Example of CPN

In CPN, each arc connecting place and transition have an arc inscription. Arc inscription is constrained by the type of its input or output place. If the source of the arc is placed, the arc inscription is constrained by the type of its input place. And if the source of the arc is a transition, the arc inscription is constrained by the type of its output place. Arc inscription can also be filled by the token time increment function. Based on this time increment function, the time information related to the token can be modified. In this research, the simulation model that we used follows CPN representation and mechanism. When we incorporate performance analysis information into the simulation model, we will put this information in the arc inscription.

3. **Simulation Framework.** This section describes the detailed framework of our application, that is, architecture, model integration, simulator, and design interface.

3.1. **Architecture.** The architecture of our simulation framework is presented in Figure 3. The core architecture consists of four main parts, namely the model integration, simulator engine, system interface (REST API), and UI engine. The core of the simulation framework is based on Python programming language. We use Redis server in this implementation to delegate long-running jobs from the flask web-server.

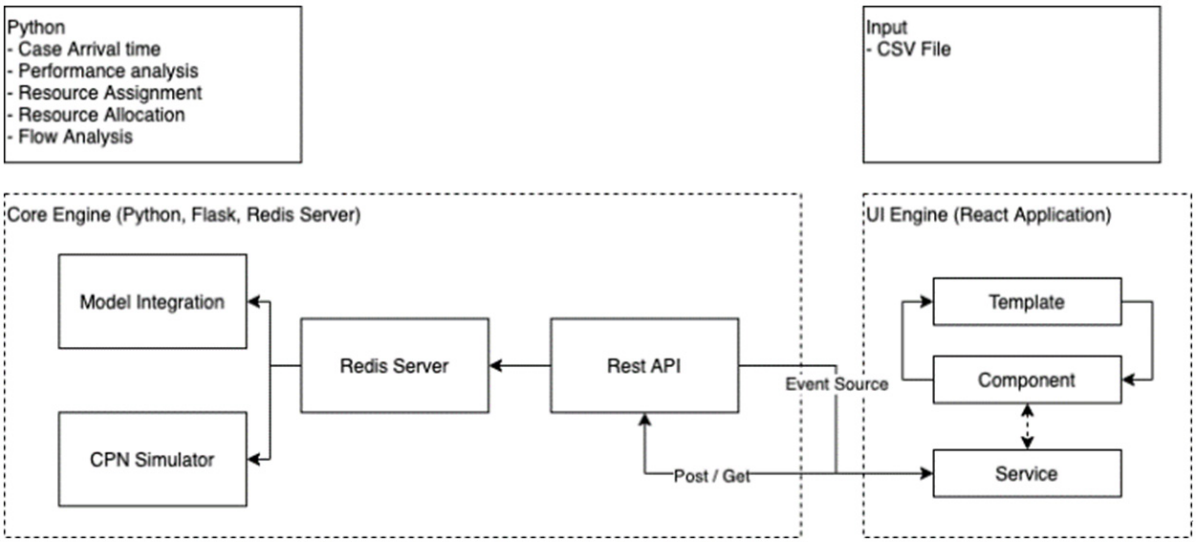

FIGURE 3. Simulation framework

3.2. **Model integration.** In the initial stage of the simulation, a user needs to upload the dataset, which is CSV file that will be used in the simulation. After the dataset is uploaded, the user can generate the flow of simulation by initiating the flow analysis module. This module is the backbone of the simulation model because it determines the flow of the simulation. When the flow analysis has already been conducted, we can complete other modules needed for the simulation, such as the arrival of cases, and performance analysis (processing time, waiting time). If all the modules have already been generated, we have to integrate all the modules needed to generate the simulation model object.

The default performance analysis used to determine modules related to time in this application is based on the Kolmogorov Smirnov Test (KS-Test), which is one of the fitness tests to find the appropriate pure probability distribution from the data. If a user wants to use another method such as Bayesian statistics to estimate distribution using mixed probability, they can upload their own object instance in a pickle file. Basically, if we want to perform model integration in this application, all the modules resulting in the previous explanation will be transformed into String, as shown in Figure 4. We need to use String representation because the application should be able to incorporate custom code that the user could be added to the module. Our cloud application uses this kind of mechanism by imitating the Jupyter notebook mechanism when executing the code.

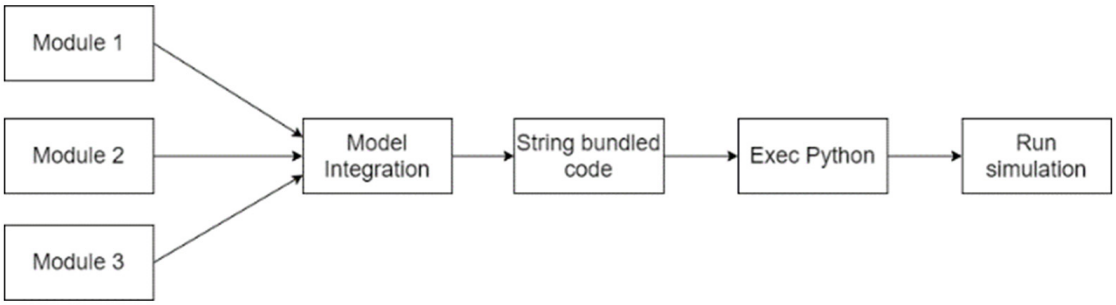

FIGURE 4. Step to model integration

3.3. **CPN simulator.** Conducting the simulation, we use CPN representation in our model that consists of place and transition with a coloured token (token with information inside). Although in the flow analysis module, there is an option to generate the flow structure based on Petri Net or just activity flow (transition without place), when we

execute the simulation, the model representation is based on CPN. In the model integration, we convert the flow (if still in activity flow) into Petri Net representation and add token information in the place. The simulator work by analyzing enabled transition. If there is more than one transition is enabled at a time, then the order of execution for that transition is conducted in an arbitrary manner (randomly), except some priority index is defined in some activities. The engine will choose the transition with the highest priority index.

The simulation will be run until the condition of stopping criteria is reached. The stopping criteria can be based on the maximum number of tokens generated, the maximum global time simulation, or any condition set by the user.

3.4. **UI engine.** The user interface engine that we used here is based on CoreUI-React framework. We used CoreUI-React framework because of its simplicity, reusable components, its ability to provide rich user interfaces interaction.

4. **Simulation Running.** In this research, we run the simulation based on the event logs dataset of container terminal operation. The step below with the picture explains the step to conduct the simulation.

- In the first step, we should upload the dataset and map the column dataset into an event log that can be processed by process mining technique, as shown in Figure 5 and Figure 6.

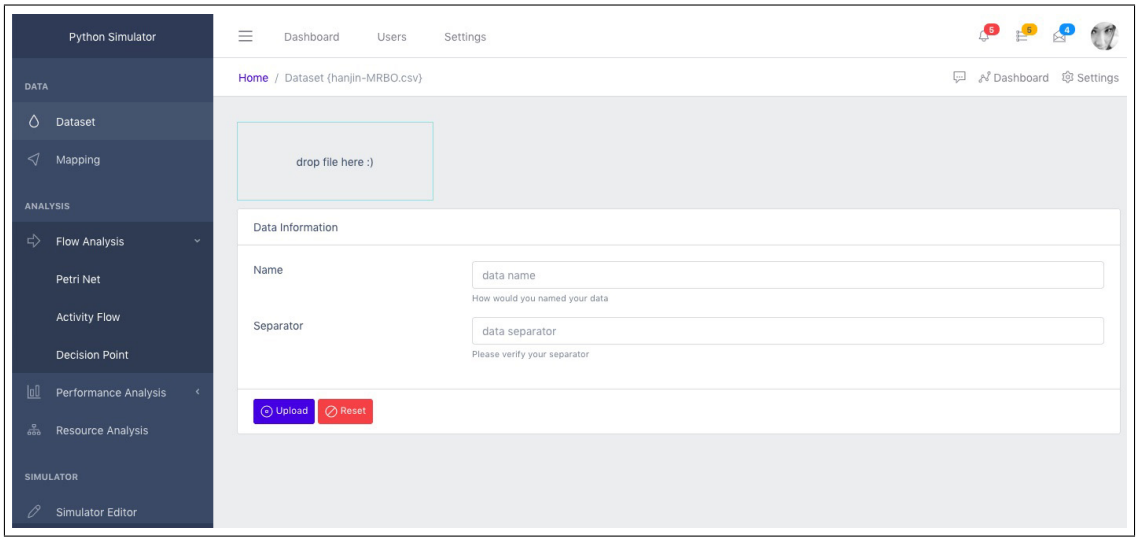

FIGURE 5. Page to upload the dataset

- After mapping the dataset attributes, we can generate the process model, as shown in Figure 7.
- After generating the process model, we can continue to run another module to analyze the performances.
- Customize the model if you want by adding more activity or other properties in the model. Then if you are already satisfied with the model, you can just run the simulation, as shown in Figure 8.

5. **Results.** We generated the simulation model based on the previous module explained in the simulation framework. We run the simulation several times in different settings of an experiment. We used the default setting of the experiment generated from the module and also changed the setting to analyze the effect of the simulation result. The result is shown in Table 1.

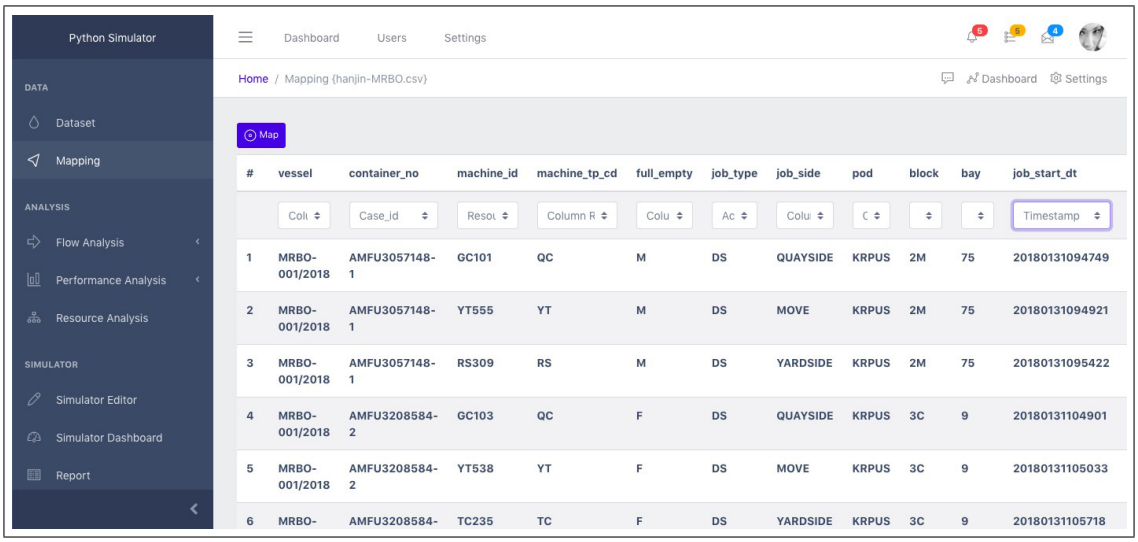

FIGURE 6. Page to map the dataset into the event log

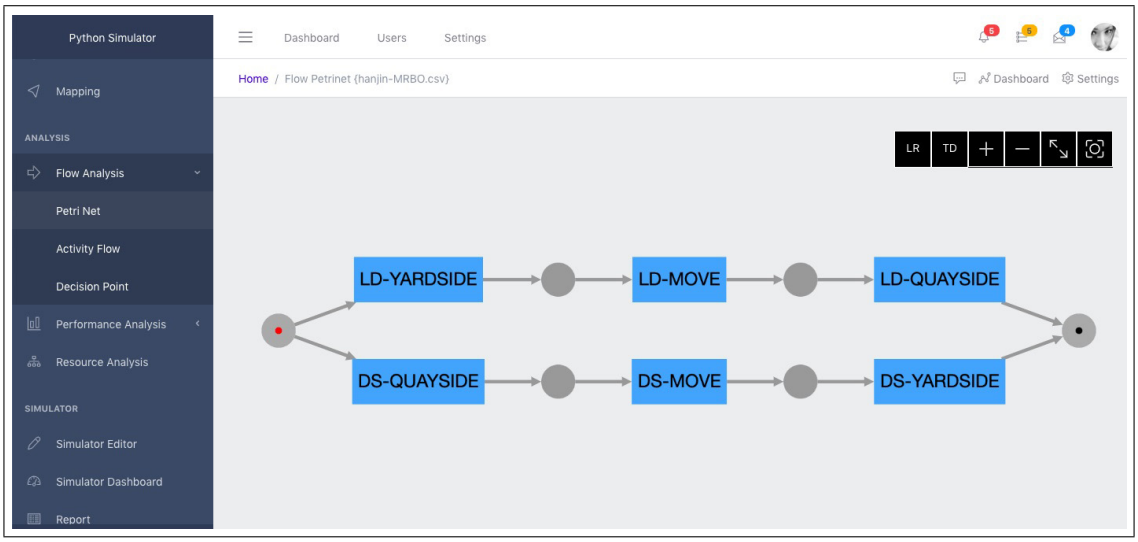

FIGURE 7. Page to display the flow analysis

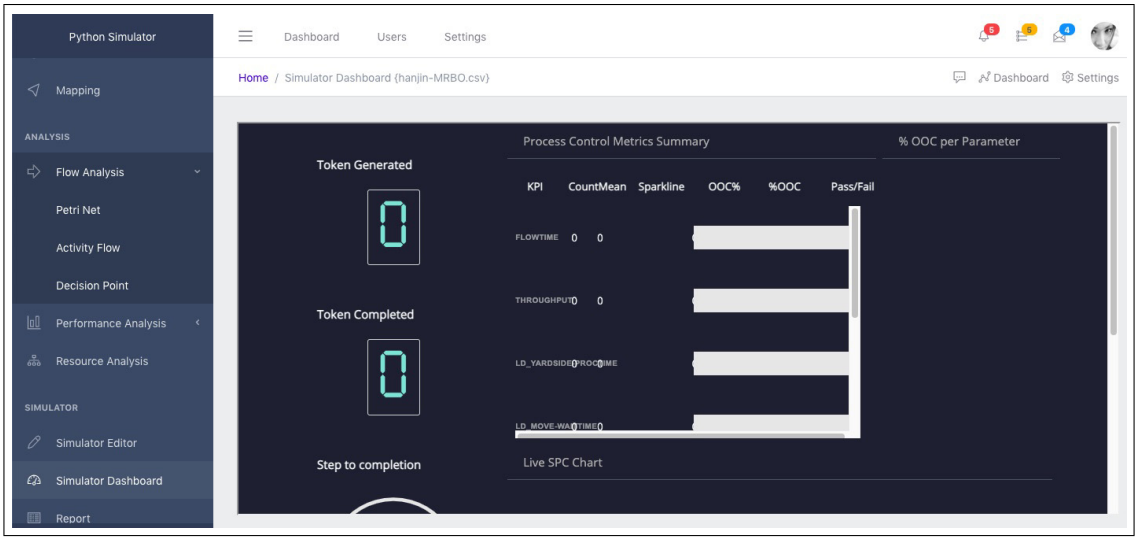

FIGURE 8. Simulation dashboard

TABLE 1. Experimental result of the simulation model

Setting	GC	YC	Container finished
1	7	94	3818
2	7	84	3856
3	7	104	3913
4	6	94	3303

Setting 1 in this experiment is the default setting that we used from the generated module. We use stopping criteria using the same time duration that we retrieved from the event logs, which is 20 hours and 46 minutes. The baseline container finished using the same configuration of resources in our event logs is 4021 containers. Based on the experiment, we got the result not far from the setting in experiment 1, with the deviation of only about 5%. Setting 2 and 3 in this experiment changes the configuration of resource YC to 84, and 104 generated number of containers finished not far from the setting 1. This could be indicated that the activity that uses resource YC is not the bottleneck activity. Setting 4 which changes the configuration of resource GC to 6, shows the number of containers finished quite far from the default setting decreasing to 3303, which indicated the activity that uses resource GC is the bottleneck activity.

6. **Conclusion.** In this paper, we were conducting research to implement an automatic generated simulation model on the cloud with python. We use a flexible module in this implementation, so the module used by the simulation model can be plugged and unplugged. Implementing this mechanism, we use a pickle file that can store instance objects in python. Although using the modular module seems reasonable, the only problem with this kind of implementation is related to security. When the application loads the pickle file, it will run the object without considering if the code is safe or not.

Analyzing the experimental result, we can see that it is possible to implement an automatic generated simulation model from an event log. However, we use the assumption that event logs used as an input here represent the real operation. Our future work for this research is to implement the deep learning mechanism in the resource allocation module of the simulation model generated from the event log.

Acknowledgment. This research was a part of the project titled 'Smart Port IoT Convergence and Operation Technology Development', funded by the Ministry of Oceans and Fisheries, Korea.

REFERENCES

[1] M. Camargo, M. Dumas and O. G. Rojas, Automated discovery of business process simulation models from event logs, *Decision Support Systems*, vol.134, 2020.

[2] M. Camargo, M. Dumas and O. G. Rojas, Learning accurate business process simulation models from event logs via automated process discovery and deep learning, *arXiv Preprint*, arXiv: 2103.11944, 2021.

[3] S. Robinson, Discrete-event simulation: From the pioneers to present, what next?, *Journal of Operational Research*, vol.56, pp.619-629, 2005.

[4] W. M. P. van der Aalst, *Process Mining: Data Science in Action*, 2nd Edition, Springer, Heidelberg, 2016.

[5] W. M. P. van der Aalst, Process mining and simulation: A match made in heaven!, *Proc. of the 50th Computer Simulation Conference*, San Diego, CA, USA, 2018.

[6] A. Rozinat, R. S. Mans, M. Song and W. M. P. van der Aalst, Discovering simulation models, *Information System*, vol.34, pp.305-327, 2009.

[7] W. M. P. van der Aalst, T. Weijters and L. Maruster, Workflow mining: Discovering process models from event logs, *IEEE Trans. Knowledge and Data Engineering*, vol.16, pp.1128-1142, 2004.

[8] A. Rozinat and W. M. P. van der Aalst, Decision mining in ProM, in *Business Process Management. BPM 2006. Lecture Notes in Computer Science*, S. Dustdar, J. L. Fiadeiro and A. P. Sheth (eds.), Berlin, Heidelberg, Springer, DOI: 10.1007/11841760_33, 2006.

[9] M. Song and W. M. P. van der Aalst, Towards comprehensive support for organizational mining, *Decision Support Systems*, vol.46, pp.300-317, 2008.

[10] I. M. Kamal, H. Bae, N. I. Utama and Y. Choi, Data pixelization for predicting completion time of events, *Neurocomputing*, vol.374, pp.64-76, 2020.

[11] S. I. Khan and A. S. M. L. Hoque, MICE: An improved missing data imputation technique, *Journal of Big Data*, vol.7, 2020.

[12] K. Okoye, S. Islam, U. Naeem and M. S. Sharif, Semantic-based process mining technique for annotation and modelling of domain process, *International Journal of Innovative Computing, Information and Control*, vol.16, no.3, pp.899-921, 2020.

[13] N. Y. Wirawan, R. A. Sutrisnowati, S. Sim, N. I. Utama, N. A. Wahid, T. N. Adi, Y. Choi, I. M. Kamal, I. R. Pulshahi and H. Bae, Petri Nets simulation for operational analytics based on process data: An overview, *ICIC Experss Letters, Part B: Applications*, vol.9, no.9, pp.1033-1040, 2018.

[14] W. M. P. van der Aalst, Process mining: Overview and opportunities, *ACM Trans. Management Information Systems*, vol.3, pp.1-17, 2012.

[15] K. Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*, 1st Edition, Springer Publishing Company, Incorporated, 2009.