PM2Sim: The Automated Creation of a Simulation Model from Process Mining

Conference Paper · October 2021

DOI: 10.1109/SMC52423.2021.9659211

2 authors, including:

Jair José Ferronato
Pontifical Catholic University of Paraná
7 PUBLICATIONS **50** CITATIONS

SEE PROFILE

2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
17-20 October, 2021. Melbourne, Australia

PM2Sim: The Automated Creation of a Simulation Model from Process Mining

Jair José Ferronato[1], Edson Emílio Scalabrin[2]

Abstract— **When used together, process mining and simulation provide better indicators for informing business management. The generation of simulation models involves significant effort, time, and system integration. The combination of process mining techniques and short-term simulation requires the extraction of real-time data from organizational systems. However, from the perspective of short-term effects and their reduced time, extraction and simulation tasks require automated execution. This study presents a framework called PM2Sim, which allows users to generate a simulation model in an automated and integrated manner with reduced effort. This framework was built on an open-source environment and event logs were used as a data source. The results obtained with PM2Sim support operational decision making and visualize scenarios of control, such as management of waiting time, process execution time, and resource capacity management.**

I. INTRODUCTION

Process mining, which is a relatively new area of research, is closely related to machine learning and process modeling and analysis. The objective of process mining is to discover, monitor, and improve real processes by extracting information from event logs available in information systems [16].

The simulation technique attempts to mimic real situations in controlled environments without putting people at risk or halting a production environment. Thus, simulation facilitates the mirroring of systems and processes in a virtual environment using real data. Generally, simulations provide long-term support for assessing strategic situations or scenarios. However, in situations such as pandemics, managers need to make decisions in a short time at an operational level. Short-term simulation provides insights for optimizing short-term actions to meet demands and improve the provision of services while the business process is executed.

Simulation is a scenario analysis tool that can be combined with optimization methods to establish an optimal solution to a problem. Owing to the complexity of the simulation, the objective function may become difficult and expensive to evaluate. The main element of simulation is modeling, which consists of reproducing the behavior of a real system in a computer system that is subjected to the same execution conditions [3].

This study aims to create an automatic simulation model using process mining. We created a framework called PM2Sim, which integrates different facilities to extract from event logs, historical data, the current state, organizational structure, and discovery of process to create scenarios.

Our paper helps automate the creation of simulation models. In particular, we focus on extracting parameters from event logs using process mining. The current state of the instances of processes in progress are inserted at the beginning of the execution of the simulation model. In addition, our work is unique because it showed with the PM2Sim framework the use of instances running processes to support operational decision-making under the perspective of detection of delays, prediction of remaining times of the instances until its completion and the recommendation of actions with the contribution of multi-criteria method.

The main contribution of this paper is the automated creation of the simulation model. The simulation model obtained with real data obtained by process mining allows to produce a more realistic model and avoids process analyst bias.

The remainder of this paper is organized as follows. Section 2 details the creation of an automatic simulation model using process mining with the PM2Sim framework and its instantiate using a simple example. Section 3 presents the framework PM2Sim. Section 4 presents the creation of an automatic simulation model. Section 5 validates the simulation model. Section 6 discusses the results based on the existing literature. Section 7 presents the conclusions, learning, limitations, and opportunities.

II. SIMULATION FROM PROCESS MINING

A. Discrete Event Simulation

Simulations can be used under different approaches, such as dynamic systems, agents, or discrete event simulation (DES). DES consists of modeling each operation of the system as a sequence of events. Each event occurs at a certain time and denotes a change in state in the system. In a process-oriented environment, DES allows the analysis of a real business process in a simulated environment as events occur, at discrete times, and on a regular basis.

Our proposal is based on the use of open-source tools that facilitate the replication of contributions in the research community in a quick and functional way. The Python language, owing to its ease of use, has been adopted by many researchers in recent years [2]. Simpy [14] is a Python library for DES that uses an object-oriented paradigm. Simpy uses generating functions to simulate processes, activities, message exchange, control the use of simple or shared resources, use of queues, and real-time simulation. Moreover,

[1]Jair José Ferronato is PhD student at the Pontifícia Universidade Católica do Paraná, R. Imac. Conceição, 1155, Prado Velho, Curitiba, PR, Brazil jair.ferronato@ppgia.pucpr.br

[2]Edson Emilio Scalabrin, Member IEEE is professor at the Pontifícia Universidade Católica do Paraná, R. Imac. Conceição, 1155, Prado Velho, Curitiba, PR, Brazil scalabrin@ppgia.pucpr.br

Simpy allows the monitoring of traces and generates statistics and logs at the end of the simulation.

The simulation allows the mining process to make predictions, thus enabling the exploration of business process scenarios. The relationship between process mining and simulation has many benefits and is considered a match made in heaven! [18].

B. Short-term Simulation

Short-term simulations explore information that is readily available in information systems. They support decisions that affect business processes in the short term and can be assessed without the need for extra modeling. Thus, a short-term simulation is used to support operational decisions. Knowledge of the initial state is essential for putting a short-term simulation into practice. Moreover, the short term represents the current state of the current target system [11]. The starting point of a short-term simulation is the current state of a business process, because in a real environment, the dynamics of a process change continuously. Thus, when starting a simulation, the actual state of the system, along with the resources allocated for the execution of certain tasks, should be considered.

The approach developed by Rozinat [13] contemplates short-term simulation, where the main contribution is the extraction of the current state from a workflow system using a computer program written in Java, and extracted according to the parameters of event logs. Historical data of events, business process model features, and the organizational model were extracted from the YAWL[1] workflow and combined to construct a simulation model. After running the simulation with CPN Tools[2], data were stored in files to compose logs of simulated events and the baseline for comparison with the original logs. This comparison is essential for validating the simulation model. One limitation of this study is that this method does not ensure the full and functional integration of the simulation environment with data sources, information extraction resources, ProM, and YAWL. Other limitations of Rozinat include how the resources were modeled and the generation of insights resulting from the simulation of effective actions. Such limitations can be overcome by improving the extraction of features from the resources present in the event logs, modeling, resource simulation, and by providing better integration between the workflow, process mining mechanism, and simulation tool.

III. Framework PM2Sim

The PM2Sim framework integrates process mining and short-term simulations. In this study, process mining was used to extract parameters for the simulation and construction of the simulation model, and operational support was provided using a multi-criteria decision-making method. The tasks of detecting deviations, predicting times, and recommending actions require the current state of the running cases.

[1]https://yawlfoundation.github.io/
[2]http://cpntools.org

PM2Sim was developed in Python using PM4Py [4] library [1]. Additionally, the Simpy framework was used for simulating discrete events. The activities and steps used in construction of PM2Sim framework are shown in Fig. 1.

The steps used to extract parameters from process mining to construct the automatic simulation model are described below.

1) **Source Data**: In this step, the event log (L_0) is extracted in XES format.
2) **Process Mining**: Different extractions are made by considering event logs as inputs. This includes: a) statistical data, such as intervals between arrivals and the average execution times of cases, b) the current state of the identification of incomplete cases, c) a process model, which involves soundness verification, and d) an organizational model with the allocated resources and their functions. The main features of the extractions were created based on the features of PM4Py [1].
3) **Expert Iteration**: The expert can intervene in this process at two moments: a) validation of the discovered process model, and b) definition of the criteria and alternatives of the multi-criteria decision method.
4) **Process Simulation**: The main part of the simulation stage was defined after the creation of the simulation model. Process mining generates a process tree [15]. This tree structure reproduced the process variants and replicated the paths for the simulation model. The probability distributions were obtained from the L_0 event log to ensure that the simulation model represented the reality better. The aim was to create new instances of the process using arrival intervals by choosing explicit decision operations for the target process. The probability distribution follows a Poisson distribution: The DES model was generated and executed using the Simpy. The model was validated by verifying the conformity between the original event log L_0 and the simulated event log L_S obtained at the end of the simulation. The statistical results of the

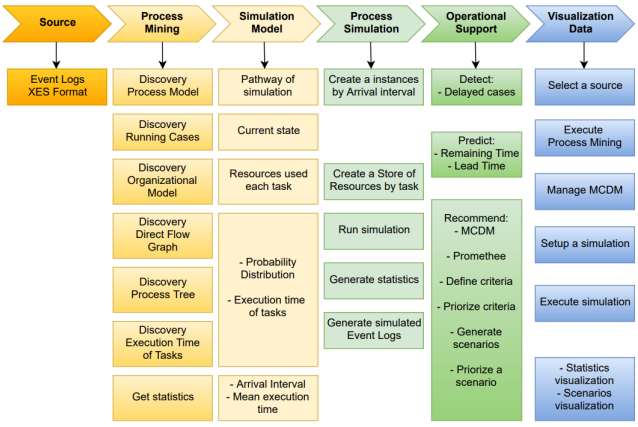

Fig. 1: Steps for constructing the PM2Sim framework

simulation were used during the operational support stage.

5) **Operational Support**: Operational support is categorized into three phases, namely: a) **Detect**: the running instances are analyzed according to the expected values of previously established KPIs. This approach allows comparing the instances that are late with the average of instances; b) **Predict**: the running instances are analyzed according to the average execution time of activities. In this way, it is possible to predict how long a given instance will finalize an execution, as well as calculate the lead time for each one built in execution; c) **Recommend**: The recommendation is made using a multi-criteria decision-making method by establishing weights for a set of defined criteria. These weights are obtained from a set of preferences related to the problem domain informed by an expert in the target business process. These preferences were compared with the statistical results of the simulation. The simulation scenarios were generated by considering the priorities given by the metrics established using a multi-criteria decision method.
6) **Visualization Data**: The results of the simulation and operational support are used for analysis and to support operational decision making.

IV. Creating an automatic simulation model

Simulation combined with process mining is a powerful tool for analysis. The following are the mechanisms used in the automatic creation of the simulation model:

A. Discovery process model

A directly follows graph (DFG) can be derived from an event log and describes what activities follow one another directly, and with which activities a trace starts or ends. In a Fig. 2, there is an edge from activity first consult to activity blood test if the first consult is followed directly by a blood test [7].

The weight of an edge denotes the frequency of occurrence. For instance, the DFG graph of our example denotes that the frequency of the pathways identified by the arches from the second consultation until the final consultation, that is, 72 for **Medicine** and 8 for **Surgery**. To obtain the frequency in DFG, we counted the number of times the source activity was followed by the target activity. These frequencies were used to create the simulation model.

B. Process Tree

The mechanism for creating a simulation model from process mining starts by reading an event log. The format of the adopted event log is XES[3]. All functionalities used in parameter extraction through process mining were performed using PM4Py.

```
log = xes_importer.apply('fileEventLog.xes')
```

[3]http://www.xes-standard.org

We extracted the process tree from the outcome of the application of the inductive miner (IM) algorithm implemented in PM4Py. The IM approach allows the discovery of sound and adequate process models in a finite time [6].The function used was:

```
tree = inductive_miner.apply_tree(log)
```

The resulting regular expression extracted from event logs is shown below:

seq ('First consult', and ('Blood test', 'X-ray scan', 'Physical test'), 'Second consult', xor ('Medicine', 'Surgery'), 'Final consult').

The symbols extracted and their representations: "**seq**" sequences, "**and**" parallelism, "**xor**" exclusive decision. From the structure of the process tree, it is possible to infer the minimum length of a trace allowed by the process model, starting with the bottom-up approach. A representation of the process tree is presented in Fig. 3.

C. Extract parameters from event logs

Extracting parameters from event logs is critical when creating automated simulation models. The basic idea is to provide data for the simulation model in an automated manner without human interference or bias. Another factor considered is the uninterrupted updating of the simulation parameters. If the data source is updated, the parameters are reflected in the simulation model.

The following are the main parameters extracted from the event logs to create the simulation model:

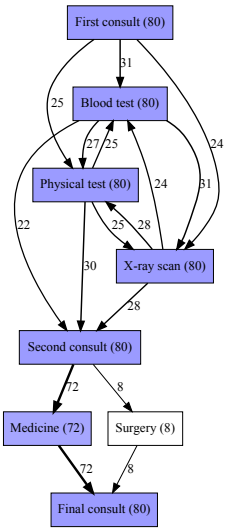

Fig. 2: Direct Flow Graph

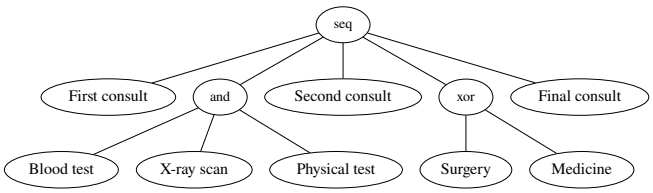

Fig. 3: Process Tree

Arrival rate: From an event log set, we retrieve the case arrival ratio, which is the average distance between the arrival of two consecutive cases in the log.

```
arrival_rate = case_arrival.get_case_arrival_avg(
    log,parameters={case_arrival.Parameters.
    TIMESTAMP_KEY: "time:timestamp"})
```

Case duration: From event logs, it is possible to calculate different statistics such as the median case duration.

```
case_duration = case_statistics.
    get_median_caseduration(p_log, parameters={
    case_statistics.Parameters.TIMESTAMP_KEY:"time
    :timestamp"})
```

Dispersion rate: From an event log set, we retrieve the case dispersion ratio, which is the average distance between the finishing of two consecutive cases in the log.

```
dispersion_rate = case_arrival.
    get_case_dispersion_avg(log, parameters={
    case_arrival.Parameters.TIMESTAMP_KEY:"time:
    timestamp"})
```

Extracting the **arrival rate** of the cases present in the event log allows this parameter to be quickly inserted into the simulation model. These data were used as input for the arrival interval of new instances in the simulation model.

Data extracted from the **case duration** were used in the model for the time simulated cases. In this way, it is possible to use the statistical information obtained in the event log to simulate the time that a case will last on average.

The **dispersion ratio** parameter was used in the validation phase of the model. The dispersion rate of the original event log was compared with that of the simulated event log.

After extracting basic parameters, such as intervals between arrivals and the average execution time of cases from event logs, the running sequences were established based on the process tree under which the simulation was processed.

Generally, probability distributions follow a Poisson distribution (the time between two arrivals is sampled following a negative exponential distribution) [5]. The probability distribution for decisions related to operations follows the frequency of each transition in the DFG.

D. Extract organizational model

The extraction of the roles of each resource present in the event log was performed using the function:

```
roles = roles_discovery.apply(log)
```

The identification of the resources from event logs that perform the activities is shown in Algorithm 1. The activities were extracted in lines 2 and 3, and for each activity, the resources that performed these activities (lines 4 to 9) were obtained. Finally, for each activity, the resources that perform that activity are stored in the dictionary.

E. Extract current state from event logs

The current state of the event log is critical for short-term simulations. We use cases that have not been completed to determine the cases that are still running. The simulation model did not start to empty. The current state of the process was inserted in the initial state of the simulation model.

Algorithm 1: Extracting organizational model

Result: Dictionary of resources by activity

1 $dic\_resources \leftarrow \{\}$
2 $activities \leftarrow attributes\_filter.get\_attribute\_values(log, "concept:name")$
3 **for** res ***in*** $activities$ **do**
4 $\quad tracefilter \leftarrow attributes\_filter.apply\_events(log, res, parameters)$
5 $\quad resources \leftarrow attributes\_filter.get\_attribute\_values(tracefilter, "org:resource")$
6 **end**
7 $dic\_resources.update(res, resources.keys())$

The parameters required for the simulation and the current state of the cases in progress were extracted via process mining. It should be noted that the discovered process model can be endorsed by an expert. The generic simulation model was automated using Algorithm 2.

Algorithm 2: Extracting current state

Result: List of running cases

1 $net, im \leftarrow inductive\_miner.apply(log)$
2 $activities \leftarrow marking\_flow\_petri(net, im)$
3 $end\_act \leftarrow end\_activities\_filter.get\_end\_activities(log)$
4 $dic\_cur\_state \leftarrow \{\}$
5 **for** act ***in*** $activities$ **do**
6 $\quad$ **if** $act\ not\ in\ end\_act$ **then**
7 $\quad\quad dic\_cur\_state.update(act, activities.keys())$
8 $\quad$ **end**
9 **end**
10 $log\_cur\_state \leftarrow pm4py.filter\_end\_activities(log, dic\_cur\_state)$
11 **for** $case$ ***in*** $enumerate(log\_cur\_state)$ **do**
12 $\quad$ **for** $event$ ***in*** $enumerate(case)$ **do**
13 $\quad\quad act\_running \leftarrow event["concept:name"]$
14 $\quad\quad res\_running \leftarrow event["org:resource"]$
15 $\quad$ **end**
16 **end**

The steps used to extract the current state of the event logs were as follows: a) discovery of the petri net and the initial state of the log at line 1; b) discovery of the activities of the process at line 2; c) filtering of the final activities of the process: this function establishes which activities have the final marking in the petri net "sink" (lines 3 and 4); d) creation and updating of the data dictionary: contains the activities and their current states. If they do not belong to the final activities, they are stored in the dictionary (lines 5 to 10). e) Filtering of log events from the current state dictionary (lines 11 and 12); f) extraction of current state data: The names of the activities and resources in use are extracted (lines 13 to 18).

The current state obtained from process mining allows for operational support to be performed. The following operations can be performed from the cases in progress:

Deviation detect: Current cases provide grants for the following deviation detection activities. For example, it is possible to measure the time spent by the case from the

beginning of the process to the current state. Obtaining throughput allows us to compare whether the time of the case in progress is close to the average of the overall execution time of the cases. If it is above average, it can generate alerts for action to be taken.

Time prediction: Time prediction can be obtained by operational support and allows you to obtain predictive estimates for the time required that a case will finish. The average of the remaining activity times was obtained, and a comparison was performed to predict from the current time of the case until its completion.

Recommend actions: The recommendation of actions is carried out with the learning of the model from the historical data. Operational support should know the decision space, that is, what possible actions can be chosen. The model obtained based on DFG contains the frequencies at which connections between activities occur. From this information, it is possible to recommend which action can be performed, with a percentage of certainty.

F. Creating a simulation model by Python Simpy

SimPy is a simulation framework interpreted using Python, which provides resources for discrete event simulation. Simpy allows one to start processes using generators, which act autonomously with the possibility of interacting through interruptions. Algorithm 3 shows the steps necessary for the automatic creation of the simulation model with Simpy.

Algorithm 3: Creating a simulation model

```
   Result: Automated simulation model
1  Function runModel():
2      for job in process_Sequence do
3          process_Instance(job)
4          audit(job)
5          generate_Results(job)
6          view_Results(job)
7      end
8  End Function
9  Function process_Instance(job):
10     while true do
11         run_Activity(job)
12         next ← random.expovariate(1/arrival)
13     end
14 End Function
15 Function run_Activity(job):
16     job.request as req:
17         yield req
18         yield timeout(random.expovariate(
19           1/job.execTime)
20     service ← random.expovariate(1/actv_Time)
21     utilization ← random.expovariate(
22       (1/arrival)/(1/service))
23 End Function
```

This excerpt of Algorithm 3 illustrates the main parts of the simulation. It is noteworthy that the *yield* function in the Simpy simulator is a Python-generating function that generates new sequences every time it is executed. The function $audit(job)$ on line 4 collects the data from the simulation. The $generate\_Results(job)$ function on line 5 performs the calculation of the indicators. The $view\_Results(job)$ function on line 6 allows us to visualize the results of the simulation. The $yield\ req$ function on line 17 requires the use of a resource, whereas, the $yield\ timeout$ function on line 18 establishes the time limit for resource allocation.

In PM2Sim, a simplified store resource was used to store the number of resources for a specific activity, such as four resources for surgery activity. The simulation results are stored in a CSV file. Pandas [8] library facilitates the filtering and analysis of simulation data.

Furthermore, other functions, such as: a) audit execution at each time interval, while simulation events continue running. b) Generation of global results for each simulated instance; c) visualization of results in graphs and texts; and d) recording of simulation-related log files in XES format.

V. VALIDATING THE SIMULATION MODEL

It should be noted that the simulation model was validated by comparing the logs generated in the simulation with the original or real logs. A comparison of the resource usage was performed. Fig. 4 shows the original log and Fig. 5 shows the simulated log. We can see that the results obtained from the utilization of resources by the simulation model are approximated to the original model.

Resource	Frequency	Relative frequency
Dr. Quinn	3	0.44 %
Dr. Charlie	5	0.73 %
Dr. Alex	5	0.73 %
Dr. Rudy	7	1.02 %
Team 1	40	5.82 %
Nurse Corey	43	6.26 %
Dr. Ben	50	7.28 %
Team 2	50	7.28 %
Nurse Jesse	57	8.3 %
Pharmacy	79	11.5 %
Dr. Bob	92	13.39 %
Lab	100	14.56 %
Dr. Anna	156	22.71 %

Fig. 4: Original log

Resource	Frequency	Relative frequency
Dr. Quinn	4	0.32 %
Dr. Alex	12	0.96 %
Dr. Charlie	29	2.32 %
Team 2	40	3.19 %
Team 1	58	4.63 %
Dr. Ben	63	5.03 %
Pharmacy	84	6.71 %
Nurse Corey	100	7.99 %
Nurse Jesse	136	10.86 %
Dr. Bob	211	16.85 %
Lab	236	18.85 %
Dr. Anna	279	22.28 %

Fig. 5: Simulated log

Using the XES file facilitates the process of validating the simulation model against the discovered process model. The Fig. 6 shows the comparison of the arrival and dispersion rates of instances present in the logs and the average duration of cases. It can be observed that the average duration of the original cases is 13974.25 min and the average duration for the simulated logs is 10500.10 minutes. The dispersion rate of the system, that is, the interval between two simulation

outputs, is equivalent; thus, we can infer that the simulation model created is valid.

```
Statistics of  Original  event log
Case arrival ratio:  1977.7474747474746  minutes
Case dispersion ratio:  2106.9380471380473  minutes
Median case duration:  13974.25  minutes
Statistics of  Simulated  event log
Case arrival ratio:  1195.885030952381  minutes
Case dispersion ratio:  2106.9380471380473  minutes
Median case duration:  10500.105783333333  minutes
```

Fig. 6: Statistics - original vs simulated

VI. DISCUSSION

The main contributions of PM2Sim are:

- automatic creation of the simulation model based on the process tree — obtained by discovering the model of process instances;
- extraction of the current state and take into account of each incomplete case to provide operational support for the detection of deviations, prediction of remaining times, and recommendation of scenarios;
- standardization of the data source — XES format. Such standardization allows the simulation to be conducted with low effort and provides operational support; and
- definition of a flow for short-term simulation by iterating the data source, process mining, simulation, operational support, and operationally such flow with computational tools, i.e., integrating the needs described in [13] in a single operational environment.

The automatic discovery of the simulation model by [12] was performed using the CPN Tools tool. However, to create the simulation model with CPN tools, knowledge of the standard machine language (SML) as described in [10] is required to define functions and capture the output as an event log. CPN tools are not an open source and do not receive frequent updates to your project [9].

The Python Simpy framework proved to be a versatile, extensible, open-source DES tool with an active community to update the framework and exchange knowledge.

The use of process mining techniques allows the creation of more reliable simulation models. Moreover, the simulation can be transformed into a powerful tool for operational decision-making using real-time process data [17].

VII. CONCLUSIONS

In this study, we presented PM2Sim, a simulation framework to automate the creation of a simulation model from process mining. PM2Sim allows the extraction of parameters from event logs the current state of instances of process to automatize support operational systems in the process environment.

The PM4Py framework was used to process mining and obtain the parameters used to create the simulation model. Python Simpy framework was used to create and execute a simulation model.

The simulation model was established based on the process tree and the direct flow graph. The simulated sequences were established using the probability distributions of possible paths for the execution of the simulated cases.

One limitation of this study is the complexity of the discovered process model. We observed a behavior consistent with the real system for a process with up to 20 activities.

In future studies, we intend to expand the generalization of PM2Sim to include other application domains, such as manufacturing and services. We also intend to explore the operational support system and illustrate the cases graphically in real time.

ACKNOWLEDGMENT

The authors would like to thank the support and funding of this research by CAPES Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 under Grant Nos.: 88887.341518/2019-00

REFERENCES

[1] A. Berti, S. J. van Zelst, and W. van der Aalst, "Process Mining for Python (PM4Py): Bridging the Gap Between Process- and Data Science," *CEUR Workshop Proceedings*, vol. 2374, pp. 13–16, may 2019. [Online]. Available: http://arxiv.org/abs/1905.06169

[2] S. Cass, "Top Programming Languages 2020," 2020. [Online]. Available: https://spectrum.ieee.org/at-work/tech-careers/top-programming-language-2020

[3] L. Chwif and A. C. Medina, *Modelagem e simulação de eventos discretos. Teoria e aplicações*, 4th ed., A. C. M. Leonardo Chwif, Ed. Rio de Janeiro: Elsevier, 2015.

[4] F. I. for Applied Information Technology, "State-of-the-art-process mining in python." [Online]. Available: https://pm4py.fit.fraunhofer.de/

[5] F. S. Hillier and G. J. Lieberman, *Introduction to operations research*, 7th ed. McGraw-Hill, 2001.

[6] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs - a constructive approach," in *Application and Theory of Petri Nets and Concurrency*. Berlin, Heidelberg: Springer, 2013, pp. 311–329.

[7] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Scalable process discovery and conformance checking," *Software and Systems Modeling*, vol. 17, no. 2, pp. 599–631, 2018.

[8] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[9] F. Pommereau, "Quickly prototyping Petri nets tools with SNAKES," LACL - Université Paris Est, Tech. Rep., 2008. [Online]. Available: http://www.univ-paris12.fr/lacl/pommereau/

[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," in *Applications and Theory of Petri Nets 2003*. Springer, 2003.

[11] H. a. Reijers and W. M. P. V. D. Aalst, "Short-Term Simulation: Bridging the Gap between Operational Control and Strategic Decision Making," *Proceedings of the IASTED International Conference on Modelling and Simulation*, pp. 417–421, 1999.

[12] A. Rozinat, R. S. Mans, M. Song, and W. M. van der Aalst, "Discovering simulation models," *Information Systems*, vol. 34, no. 3, pp. 305–327, 2009.

[13] A. Rozinat, M. T. Wynn, W. M. van der Aalst, A. H. ter Hofstede, and C. J. Fidge, "Workflow simulation for operational decision support," *Data and Knowledge Engineering*, vol. 68, no. 9, pp. 834–850, 2009.

[14] Simpy, "Overview documentation - discrete event simulator." [Online]. Available: https://simpy.readthedocs.io/

[15] A. Vahedian K. and S. Alizadeh, "A new model for discovering process trees from event logs," *Applied Intelligence*, vol. 41, no. 3, 2014.

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

[17] W. M. Van Der Aalst, "Business process simulation survival guide," *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*, pp. 337–370, 2015.

[18] W. M. van der Aalst, "Process mining and simulation: A match made in heaven!" *Simulation Series*, vol. 50, no. 10, pp. 39–50, 2018.