

# A Python Extension to Simulate Petri nets in Process Mining <sup>\*</sup>

M. Pourbafrani<sup>1</sup>, Sandhya Vasudevan<sup>2</sup>, Faizan Zafar<sup>2</sup>, Yuan Xingran<sup>2</sup>,  
Ravikumar Singh<sup>2</sup> and Wil M. P. van der Aalst<sup>1</sup>

<sup>1</sup> Chair of Process and Data Science, RWTH Aachen University, Germany

{mahsa.bafrani,wvdaalst}@pads.rwth-aachen.de

<sup>2</sup> RWTH Aachen University {sandhya.vasudevan,

faizan.zafar,xingran.yuan,ravikumar.singh}@rwth-aachen.de

**Abstract.** The capability of process mining techniques in providing extensive knowledge and insights into business processes has been widely acknowledged. Process mining techniques support discovering process models as well as analyzing process performance and bottlenecks in the past executions of processes. However, process mining tends to be “backward-looking” rather than “forward-looking” techniques like simulation. For example, process improvement also requires “what-if” analyses. In this paper, we present a Python library which uses an event log to directly generate a simulated event log, with additional options for end-users to specify duration of activities and the arrival rate. Since the generated simulation model is supported by historical data (event data) and it is based on the Discrete Event Simulation (DES) technique, the generated event data is similar to the behavior of the real process.

**Keywords:** process mining, simulation, discrete event simulation, event log, automatic simulation model generation.

## 1 Introduction

Process mining tools provide unique capabilities to diagnose business processes existing within organizations (e.g., in transaction logs or audit trails) including discovering the running processes, as well as deviations and bottlenecks that occur or exist in the current state of the processes [1]. In all of the proposed tools for simulation in process mining, interaction with the user and user knowledge is an undeniable requirement for designing and running the simulation models. Moreover, most of the approaches are dependent on external simulation tools. For instance, in [2], the proposed business process simulation technique is based on the BPMN model. All the simulation parameters with the BPMN model are put into a simulation tool such as BIMP for the simulation step. [3] provides a comprehensive platform for modeling stochastic Petri nets, however, the connection to process mining is missing. In [4], the created simulation model is

---

<sup>\*</sup> Acknowledgments Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy-EXC-2023 Internet of Production – 390621612. We also thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

based on the CPN tool which requires users to have knowledge of discrete event simulation as well as *Standard Machine Language* (SML) to define functions and capture the output as an event log [5]. In [6] an external tool, i.e., ADONIS for simulating the discovered model and parameters are used. It should be noted

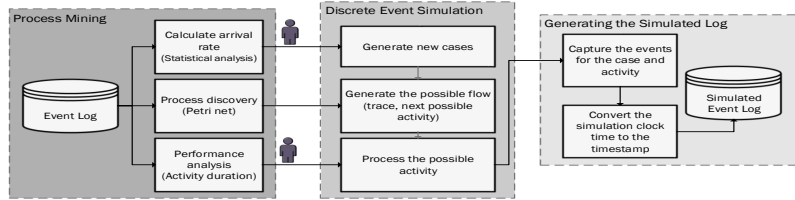


Fig. 1: The general framework for discrete event simulation in process mining. The automatic generation of simulation models and the corresponding simulated event logs is possible by starting with an event log, extracting the process model and the performance information, generating random cases, and finally converting the processed activities in the form of events. The user annotations indicate the options for the user to simulate the process with user-defined parameters.

that oftentimes the user does not need to have in-depth knowledge of the process so as to simulate it which holds for the most of commercial tools such as *Protos*, *Any Logic* and *ARENA*. For instance, when the user only needs to know how the process will behave if the average arrival rate increases to 5 minutes, i.e., every 5 minutes a new case arrives.

In process mining, the above-mentioned requirements can be addressed by the concept of Discrete Event Simulation (DES) [7]. DES for business processes has been developed in *Java* as a plugin of *ProM* [8]. However, custom options such as the ability to change the duration of activities for future performance analyses are missing. Approaches such as in [9] uses the same idea in *Java*, including some drawbacks, e.g., a fixed duration for case generation step. The generated cases do not have any time overlap, which is not the case in reality. Work such as [10] tries to generate a business simulation model for business processes which relies on the user domain knowledge. [11] describes a range of modeling tasks that should be considered when creating a realistic business process simulation model.

Existing process mining tools provide users with a visual representation of process discovery and performance analyses using event data in the form of event logs. Therefore, an approach is needed to play out reality and generate the exact behavior which makes further analyses in process mining possible. Moreover, the option to extend the library as an open-source tool is easily provided. User options to add capacity to the activities and to extend the case production for different times of the day and week can be implemented.

Research work such as [12] and [13] use aggregated simulation which is useful for what-if analyses in a high-level decision-making scenario [14]. The *PMSD* tool represents the aggregated approach and generates a simulation model at a higher level of detail [15].

In this paper, we introduce an easy-to-use open-source Python-based application that connect the provided process mining environment in Python *PM4Py*

[16] to the general simulation techniques in Python, *Simpy*<sup>3</sup>. The latter library is used for discrete event simulation and handles the required system clock in DES. The automatically designed simulation model can be configured with user-defined duration for the activities and arrival rate. The final output is an event log based on the given number of cases that can be used further for process mining analyses. The designed framework of the tool is shown in Fig. 1. It is designed on the basis of three main modules; process mining, simulation, and transformation of the generated events into an event log.

## 2 PNSIM

Event logs comprise events where each event refers to a *case* (process instance), an *activity*, a *timestamp*, and any number of additional attributes (e.g., costs, resources, etc.). A set of events forms an event log which can be used in process mining analyses. As shown in Fig. 1, our approach starts with applying process mining techniques on the original event log. Therewith a process model is discovered in the form of a Petri net which presents possible flows of activities for the cases. Subsequently, performance analyses provide the case arrival rate including the business hours and the average duration of the activities. This information makes the automatic generation of process instances based on the past executions of processes possible.

We aim to provide a simulation model and the corresponding simulated event log as close to reality as possible. To do so, we perform the following preprocessing steps in the process mining module:

- Process discovery:
  - *Maximum length of traces:* The presence of loops in the process models (Petri nets) makes the generation of long unrealistic traces possible. By identifying and replacing the maximum length of traces, we limit the possibility of the execution of unrealistic loops for the simulated cases.
- Performance analyses:
  - *Arrival rate calculation:* The business hours are considered by default in calculating the average arrival rate. Moreover, we learn the inter-arrival time distribution from the actual arrival times. The detected distribution is used in the simulation step.

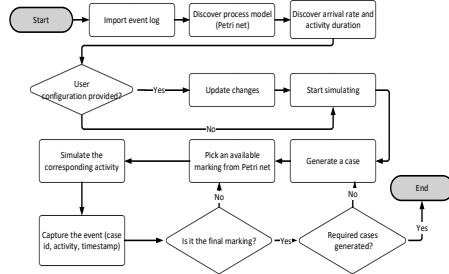


Fig. 2: The flowchart of the integrated discrete event simulation of the processes using process mining. Each activity runs if it is available and the clock of the simulation gets updated for every new event. New events are a newly arrived case, the end of processing an activity for a case, or the start of the processing of an activity for a case.

<sup>3</sup> <https://simpy.readthedocs.io>

- *Activity duration*: By removing outliers from the set of duration for each activity, we provide more robust values for the duration of activities.

Using the distribution of activities' duration, we implicitly consider the average duration of resources' time without extracting the resource pool. This aggregated calculation includes the behavior of resources for handling each activity.

Next is the simulation module in which we generate new cases. In extracting the arrival rate of cases, i.e., the duration of time for a new case to arrive, we include the business hours in the calculation of the arrival rate to obtain an accurate value. The next step is to discover how the cases are handled in the process w.r.t. the service time of each activity and the possible flow of activities that each case can take. Based on the presence of the start and complete timestamps, the value of the average duration of each activity is captured. The discovered Petri net also is used for generating a possible flow of activities. The provided user options to interact with and modify the simulation process are the following functions:

- *Activity duration* generates the random values based on the extracted values for each activity and the corresponding distribution. The user is able to change the parameters of the distribution .
- *Arrival rate* uses a normal distribution for generating new cases and the user is able to change the average arrival rate for the simulated log.
- *Case generator* produces random cases based on the provided number of cases by the user. It determines the terminating point of the simulation.

The final module is designed to transform the simulated events for the generated cases into event logs. The discrete event simulation clock is converted to the real timestamp and each activity is recorded for the cases in the real timestamp. The flow chart of the simulation module of our tool is shown in Fig. 2. After each new generated case, it checks the condition whether the number of cases provided by the user is met. Accordingly, it follows up with processing the picked marking from the Petri net. Either the provided outputs by the process mining module or user parameters are used to start the simulation. By selecting the available activity from the Petri net, the simulation module checks whether the previous process of the activity has finished. In the last step, after performing each possible event (generating a new case or processing of an activity) the simulation clock gets updated and the data is captured. Since the simulation technique considers the capacity of each activity, the concept of queuing is implicitly covered in the simulated event log. When an activity with full capacity, i.e., processing other cases, is selected for the current case, the case is in the waiting state which is shown in the performance analyses of the event log.

### 3 Tool Maturity

The source code of our tool, a tutorial, and a screen-cast are publicly available. <sup>4</sup> The tool has been used in multiple academic projects to simulate a

<sup>4</sup> <https://github.com/mbafrani/AutomaticProcessSimulation>

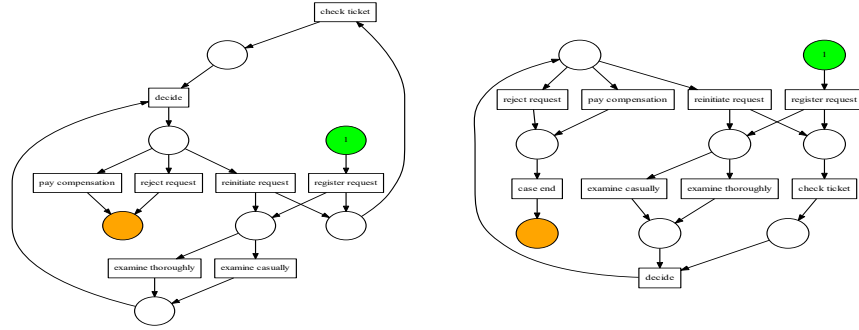


Fig. 3: The discovered process model of the example event log using Petri net notation. It includes 8 unique activities and represents the process of handling requests in an organization (a). The discovered process model of the simulated event log using Petri net notation. Our tool generates the simulated event log directly from the original event log, which captures both time and activity flow features of the original process (b).

process model in different situations and generate different event logs. For instance, for the purpose of time series analyses, different arrival rates for the same process have been selected and the tool event logs are generated. We use a sample case study to demonstrate the steps and usability of our tool.

Figure 3(a) shows a sample process model of the example event log in the form of a Petri net. We use the sample event log and simulate the process for 1000 cases. Using the same process discovery algorithm for the simulated event log result in the same model including concurrences in the model as shown in Fig. 3(b). The discovered model shows that our tool is able to mimic the process and simulate the model including the time aspects of the process. Part of the simulated log is shown in Fig. 4. The simulated event log has the main attributes of an event log. It captures the *case id* which is increased incrementally to the defined number by the user, *activity* names, and the corresponding complete time as *timestamp*.

case_id	activity	timestamp
Case 1	register request	7/2/2020 17:46:35
Case 2	register request	7/2/2020 20:46:35
Case 3	register request	7/2/2020 23:46:35
Case 4	register request	7/3/2020 2:46:35
Case 1	check ticket	7/3/2020 5:43:15
Case 5	register request	7/3/2020 5:46:35
Case 2	examine thoroughly	7/3/2020 8:43:15
Case 6	register request	7/3/2020 8:46:35
Case 3	check ticket	7/3/2020 11:43:15
Case 7	register request	7/3/2020 11:46:35

Fig. 4: Part of the simulated event log for the example event log which is generated in the .csv format. It includes the main attributes of an event log, case id, activity, and timestamp.

## 4 Conclusion

Techniques for past analyses of processes in organizations are well-supported in existing academic and commercial process mining tools. However, future analyses for business processes are not fully covered in the current tools. Commonly used options either need knowledge of simulation techniques and modeling, high interaction with users or are not accurate enough since they are not supported by real event data. In this paper, we presented the tool which directly uses the event data of a process in the form of an event log and simulates the process with the automatically extracted values as well as user-defined input. The tool

is designed to simulate the processes in different scenarios. Since the simulation module is based on the discrete event simulation technique, the simulated event log includes the same behavior as the real event log.

## References

1. W. M. P. van der Aalst, *Process mining - data science in action, Second Edition*. Springer, 2016.
2. M. Camargo, M. Dumas, and O. G. Rojas, “Simod: a tool for automated discovery of business process simulation models,” pp. 139–143, 2019.
3. S. Baarir, M. Beccuti, D. Cerotti, M. De Pierro, S. Donatelli, and G. Franceschinis, “The greatspn tool: recent enhancements,” *SIGMETRICS Performance Evaluation Review*, vol. 36, pp. 4–9, 03 2009.
4. A. Rozinat, R. S. Mans, M. Song, and W. M. P. van der Aalst, “Discovering simulation models,” *Inf. Syst.*, vol. 34, no. 3, pp. 305–327, 2009.
5. 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, 24th International Conference, ICATPN 2003, Eindhoven, The Netherlands, June 23-27, 2003, Proceedings*, pp. 450–462, 2003.
6. B. Gawin and B. Marcinkowski, “How close to reality is the as-is business process simulation model?,” *Organizacija*, vol. 48, no. 3, pp. 155 – 175, 2015.
7. W. M. P. van der Aalst, “Process mining and simulation: a match made in heaven!,” in *Proceedings of the 50th Computer Simulation Conference, SummerSim 2018, Bordeaux, France, July 09-12, 2018*, pp. 4:1–4:12, ACM, 2018.
8. B. F. van Dongen, A. K. A. de Medeiros, H. Verbeek, A. Weijters, and W. M. van der Aalst, “The ProM framework: A new era in process mining tool support,” in *International Conference on Application and Theory of Petri Nets*, pp. 444–454, Springer, 2005.
9. A. Rogge-Solti and M. Weske, “Prediction of business process durations using non-markovian stochastic Petri nets,” *Inf. Syst.*, vol. 54, pp. 1–14, 2015.
10. L. Pufahl and M. Weske, “Extensible BPMN process simulator,” in *Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM)*, 2017.
11. N. Martin, B. Depaire, and A. Caris, “The use of process mining in business process simulation model construction - structuring the field,” *Bus. Inf. Syst. Eng.*, vol. 58, no. 1, pp. 73–87, 2016.
12. M. Pourbafrani, S. J. van Zelst, and W. M. P. van der Aalst, “Scenario-based prediction of business processes using system dynamics,” in *On the Move to Meaningful Internet Systems: OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21-25, 2019, Proceedings*, pp. 422–439, 2019.
13. M. Pourbafrani, S. J. van Zelst, and W. M. P. van der Aalst, “Supporting automatic system dynamics model generation for simulation in the context of process mining,” in *Business Information Systems - 23rd International Conference, BIS 2020, Colorado Springs, CO, USA, June 8-10, 2020, Proceedings*, pp. 249–263, 2020.
14. M. Pourbafrani, S. J. van Zelst, and W. M. P. van der Aalst, “Supporting decisions in production line processes by combining process mining and system dynamics,” in *Intelligent Human Systems Integration 2020 - Proceedings of the 3rd*

- International Conference on Intelligent Human Systems Integration (IHSI 2020): Integrating People and Intelligent Systems, February 19-21, 2020, Modena, Italy*, pp. 461–467, 2020.
15. M. Pourbafrani and W. M. P. van der Aalst, “PMSD: Data-driven simulation in process mining,” in *Proceedings of the Dissertation Award, Doctoral Consortium, and Demonstration Track at BPM 2020 co-located with 18th International Conference on Business Process Management, BPM 2020*, 2020.
  16. A. Berti, S. J. van Zelst, and W. M. P. van der Aalst, “Process mining for python (pm4py): Bridging the gap between process- and data science,” *CoRR*, vol. abs/1905.06169, 2019.