

SimPN: A Python Library for Modeling and Simulating Timed, Colored Petri Nets

Remco M. Dijkman

Eindhoven University of Technology, Eindhoven, The Netherlands

Abstract

SimPN is a Python library for discrete event simulation using timed, colored Petri nets. It provides essential simulation functionality, such as visualization of runs, warm-up time, replications, and reporting, including reporting to event logs that can be used for analysis in process mining tools. In addition, it supports the integration of Python functions, enabling the implementation and performance evaluation of planning and optimization functionality within a simulated business process context. The library also accommodates modeling, simulation, and visualization of higher-level languages like the Business Process Model and Notation. The library is open-source, fully documented, and has been successfully used in a course at Eindhoven University of Technology.

Keywords

Petri Net, Simulation Library, Modeling Tool


1. Introduction


Petri nets [1] are commonly used as the mathematical underpinning for methods and techniques from the area of Business Process Management. Consequently, having tool support for modeling and analyzing Petri nets is important to the discipline. Such support facilitates the development of new methods and techniques that are based on Petri net theory. At the same time it can be used to teach Petri net theory to students. Given the increasing development of analysis methods and techniques in Python, having a Python library to support the creation of such methods is highly beneficial.

SimPN provides such a Python-based library for modeling and simulating timed, colored Petri nets. It provides advanced functionality for visualizing Petri net simulations, along with essential simulation features like replications, warm-up time, and reporting. This includes reporting in the form of event logs to allow for analysis with process mining tools. Additionally, it supports the development of simulations for higher-level modeling languages, including the Business Process Model and Notation (BPMN), leveraging the same library for visualization. The library also supports the integration of Python functions. This is especially useful if such functions are used to model (in code) planning or optimization functionality, because then such functionality can be evaluated in the context of the business process that is being simulated.

Proceedings of the Best BPM Dissertation Award, Doctoral Consortium, and Demonstrations & Resources Forum co-located with 22nd International Conference on Business Process Management (BPM 2024), Krakow, Poland, September 1st to 6th, 2024.

✉ r.m.dijkman@tue.nl (R. M. Dijkman)

ORCID  0000-0003-4083-0036 (R. M. Dijkman)

 © 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

The remainder of this paper further introduces the library and is structured as follows. Section 2 presents other Petri net modeling and simulation tools and in doing so presents the innovations that the proposed library brings. Section 3 presents the Petri net modeling and simulation features of the library, and Section 4 presents the maturity of the library.

2. Related Work

There exist a large number of Petri net modeling and simulation tools and libraries, in particular in the area of Business Process Management. The proposed library contributes to this collection by providing a Python library with full timed, colored Petri net simulation capabilities as well as native integration with Python functionality.

A recent overview of Petri net modeling and simulation tools and libraries is provided by Kumbhar and Chavan [2]. This overview covers Petri net modeling and simulation tools, among which the long-running and well-maintained tools WoPeD [3] and CPN Tools [1]. The overview also discusses Petri net libraries, including the Matlab-based library SimHPN [4] and the Python-based library SNAKES [5]. SNAKES is the closest to the proposed library. The two main differences between SimPN and SNAKES are:

- SimPN has a more comprehensive integration with Python by supporting Python functions, while SNAKES only supports Python expressions. This has important implications for the ability to use other functionality, in particular functionality for business process optimization, as will be discussed in Section 3.
- SimPN has stronger business process simulation support. In particular it provides direct support for timed Petri nets, including their visual simulation, while in SNAKES support for timed Petri nets has to be implemented separately. Also, SimPN provides additional functionality for business process simulation, including support for simulating and visualizing higher level languages, such as BPMN, and support for replications, warm-up time, and direct export to process mining tools.

These differences are due to fundamental design choices to make SimPN and SNAKES suitable for different use cases. In particular, SimPN primarily aims at supporting business process simulation and optimization, while SNAKES primarily aims at supporting formal analysis of Petri nets. This, for example, makes it logical for SimPN to support Python functions and SNAKES to just support Python expressions instead. It also makes it logical for SimPN to have more comprehensive functionality for simulation.

3. Features

The SimPN library is primarily designed for business process simulation. While it can be used as a general simulation tool, it has an emphasis on simulating planning and optimization of business processes, such as simulating a policy for allocating employees to tasks in a business process. This facilitates studies that aim to develop and evaluate business process planning and optimization functionality. To this end, the SimPN library provides the following features:

- Simulation functionality, including support for visualizing simulation runs, replications, warm-up time, reporting, and export to process mining tools.

- Support for Python functions, including support for planning and optimization through mathematical programming, heuristics, and computational intelligence.
- Higher level language support, including support for visualizing higher-level language constructs.

3.1. Simulation Functionality

The SimPN library is based on timed, colored Petri nets. As such, it has places, transitions, and tokens, where tokens can have colors and delays. The code below illustrates how a simple simulation can be set up for a shop with a cashier and two waiting customers. The cashier serves the customers with a delay of 0.75 time units.

```
shop = SimProblem()

resources = shop.add_var("resources")
customers = shop.add_var("customers")
resources.put("cashier")
customers.put("c1")
customers.put("c2")

def process(customer, resource):
    return [SimToken(resource, delay=0.75)]
shop.add_event([customers, resources], [resources], process)

shop.simulate(10, SimpleReporter())
```

The code shows how two places are created using the `add_var` method, one for the cashier resources and one for the customers. Subsequently, one cashier is added to the resources place and two customers are added to the customers place.

Transitions can be created using the `add_event` method. This method takes the list of incoming places and the list of outgoing places of the transition as parameters. In addition, a function must be defined that describes how the transition generates outgoing tokens based on incoming tokens. In the example, the `process` function takes (the value of) a token from the customers place, which it calls `customer` and (the value of) a token from the resources place, which it calls `resource`. The function must return a list of tokens that are put on the outgoing places. In this case, the function returns a single token that has the same value as the incoming resource token. The token is made available with a delay of 0.75 time units, which simulates that it takes a cashier 0.75 time units to process a customer.

The simulation is started using the `simulate` method. This method takes the number of time units to simulate and a reporter as parameters. The reporter is an object that can be used to report what happens in the simulation. In the example, the `SimpleReporter` is used, which simply prints the firing of each transition to the standard output.

The library also has more advanced reporting functionality. In particular it provides functionality for adding warm-up times and replications to obtain valid conclusions from the simulation.

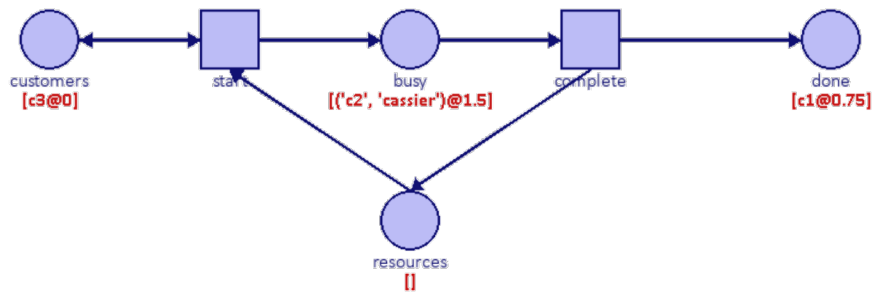


Figure 1: Visualization of a Petri net simulation.

It also provides functionality for reporting to an event log that can subsequently be used for process mining and it provides functionality for visualizing the Petri net and step-wise animation of simulation as illustrated in Figure 1.

3.2. Support for Python Functions

The Python code example above illustrates that the library has support for invoking Python functions. Any self-defined Python function can be used in the simulation. These functions can take token values as input and produce token values as outputs. Lambda functions can also be used to simplify the code, in case only simple passing of token values is required.

The benefits of using Python functions are mainly sought when evaluating business process optimization functionality through simulation. The optimization functionality can be implemented as a Python function. Subsequently, the effect of the optimization function on a particular business process can be evaluated by simulating that business process as a Petri net in combination with the optimization functionality. As an example consider the simulation of a treatment process in a hospital, where patients follow a path through through different activities and wards in the hospital that depends on their diagnosis. A planning function can be developed in Python that plans patients in timeslots upon arrival. The reception of the patient in the timeslot and subsequent progress of the patient through the hospital, depending on the availability of resources, can then be simulated as a Petri net.

In previous work we have specifically studied the integration of Petri nets with planning functions based on Deep Reinforcement Learning [6].

3.3. Higher-level Language Support

The SimPN library has support for higher-level languages of which the semantics is defined in terms of Petri nets. Some concepts from the Business Process Model and Notation (BPMN) have been implemented in the library already. Using the library, it is possible to define a higher-level language construct, like a BPMN task or choice construct in terms of Petri net places and transitions. This is done by defining a class for this construct that inherits from the `Prototype` class and generates the desired composition of places and transitions when it is instantiated. That class can then be instantiated each time the specific higher-level language construct is

required. For example, the following code creates a BPMN start event with a label ‘customer arrived’ and an exponential inter-arrival time with a mean of 1.

```
prototype.BPMNStartEvent(my_model, [], [arrived], "customer arrived",  
                          lambda: random.expovariate(1))
```

A prototype can easily be enriched with visualization functionality, which enables its simulation to be visualized according to its own notation.

4. Tool Maturity

The SimPN library is available open source¹ and can be installed using the Python package manager pip by the name `simpn`. The library is fully documented² and comes with a large number of examples that demonstrate its functionality. The documentation also contains teaching materials that explain the theory and examples in detail and a short video³ demonstrating the main functionality. These teaching materials will be further developed as the library develops. The SimPN library has successfully been used in the 2024 iteration of the course ‘Business Process Simulation’ at Eindhoven University of Technology with 39 students.

References

- [1] K. Jensen, L. M. Kristensen, L. Wells, Coloured petri nets and cpn tools for modelling and validation of concurrent systems, *Intl. Journal on Software Tools for Technology Transfer* 9 (2007) 213–254.
- [2] V. B. Kumbhar, M. S. Chavan, A review of petri net tools and recommendations, in: *Proc. of the Intl. Conf. on Applications of Machine Intelligence and Data Analytics (ICAMIDA)*, Atlantis Press, 2023, pp. 710–721.
- [3] T. Freytag, *Woped–workflow petri net designer*, University of Cooperative Education (2005) 279–282.
- [4] J. Júlvez, C. Mahulea, C.-R. Vázquez, SimHPN: A MATLAB toolbox for simulation, analysis and design with hybrid petri nets, *Nonlinear Analysis: Hybrid Systems* 6 (2012) 806–817.
- [5] F. Pommereau, SNAKES: A flexible high-level petri nets library, in: R. Devillers, A. Valmari (Eds.), *Proc. of the Intl. Conf. on Application and Theory of Petri Nets and Concurrency (PETRI NETS)*, Springer, 2015, pp. 254–265.
- [6] R. Lo Bianco, R. Dijkman, W. Nuijten, W. van Jaarsveld, Action-evolution petri nets: A framework for modeling and solving dynamic task assignment problems, in: *Prof. of Intl. Conf. on Business Process Management (BPM)*, Springer, 2023, pp. 216–231.

¹<https://github.com/bpogroup/simpn>

²<https://bpogroup.github.io/simpn/>

³https://youtu.be/YaR_M15UBKY