

# Distributed Event Factory: A Tool for Generating Event Streams on Distributed Data Sources

Hendrik Reiter<sup>1,\*</sup>, Christian Imenkamp<sup>2</sup>, Agnes Koschmider<sup>2</sup> and Wilhelm Hasselbring<sup>1</sup>

<sup>1</sup>Christian-Albrechts-University Kiel, Christian-Albrechts-Platz 4, 24118 Kiel, Germany

<sup>2</sup>University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany

## Abstract

In real-life applications, data sources are often distributed. In a smart factory, data is generated by spatially distributed sensors. Distributed process mining algorithms may exploit this data locality by processing data *where* it is generated. The Distributed Event Factory is a tool to evaluate distributed process mining algorithms under (best-effort) realistic conditions. It generates synthetic event streams that consider the distributed nature of the data sources. In particular, we can evaluate the scalability of such algorithms by increasing the volume and velocity of the generated events. Additionally, other external factors such the temporal behavior of events, and varying load profiles can be configured. Using the example of a smart factory, we demonstrate the tool's capabilities.

## Keywords

Event Log Generator, Stream Process Mining, Distributed Process Mining, Distributed Computing, Markov Chain

Metadata description	Value
Tool name	Distributed Event Factory (DEF)
Current version	0.1.0
Legal code license	Apache 2.0
Languages, tools and services used	Python, Kafka
Supported operating environment	GNU/Linux, MacOS
Download/Demo URL	<a href="https://github.com/cau-se/DistributedEventFactory/releases/tag/0.1.0">https://github.com/cau-se/DistributedEventFactory/releases/tag/0.1.0</a>
Documentation URL	<a href="https://cau-se.github.io/DistributedEventFactory/">https://cau-se.github.io/DistributedEventFactory/</a>
Source code repository	<a href="https://github.com/cau-se/DistributedEventFactory">https://github.com/cau-se/DistributedEventFactory</a>
Screencast video	<a href="https://youtu.be/r2RzP9DwOqk">https://youtu.be/r2RzP9DwOqk</a>

## 1. Introduction

Process mining is a discipline dedicated to discovering and monitoring real-world processes. Streaming process mining [1] aims to deliver the results of process mining algorithms as soon as

ICPM 2024 Tool Demonstration Track, October 14-18, 2024, Kongens Lyngby, Denmark

\*Corresponding author.

✉ [hendrik.reiter@email.uni-kiel.de](mailto:hendrik.reiter@email.uni-kiel.de) (H. Reiter); [christian.imenkamp@uni-bayreuth.de](mailto:christian.imenkamp@uni-bayreuth.de) (C. Imenkamp); [agnes.koschmider@uni-bayreuth.de](mailto:agnes.koschmider@uni-bayreuth.de) (A. Koschmider); [hasselbring@email.uni-kiel.de](mailto:hasselbring@email.uni-kiel.de) (W. Hasselbring)

🆔 0009-0003-8544-0012 (H. Reiter); 0009-0007-4295-1268 (C. Imenkamp); 0000-0001-8206-7636 (A. Koschmider); 0000-0001-6625-4335 (W. Hasselbring)



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

data is generated. The emerging field of distributed process mining further considers the spatial context of data. Distributed Process Mining Algorithms (like e.g. [2]) do not only process data when it is generated but also at the location where it originates. By processing data directly at its source, there is no need to send it to a central instance for processing. In a smart factory with multiple production facilities, the use of distributed process mining algorithms might be beneficial. Data is initially processed individually in each factory, and only the relevant data required for process analysis is forwarded to other facilities. By applying the principle of data sparsity, latencies can be minimized, privacy preserved, and network costs reduced.

What is lacking for the efficient development of distributed process mining algorithms is a tool that generates data while addressing the aspects of real-time processing and data distribution. Furthermore, it would be beneficial if this tool could also consider other issues that occur in the real world, such as noise, varying volume and velocity of data generation, and the delayed arrival of data. In process mining, there are various tools that deal with the generation of synthetic data. These either focus on privacy [3], concept drift [4] or sensor events [5]. PLG2 [6] comes closest to these requirements. It can generate messages in real time. Additionally, parameters such as noise and various model complexities can be configured. However, data distribution is not sufficiently considered. Furthermore, sending rates cannot be adjusted arbitrarily, and temporal dependencies of data cannot be configured. An alternative approach calls for considering properties of event logs and including (data) distribution into it. This requires to incorporate additional properties into event logs like the reference of the location. Additionally, event logs are finite and cannot be used to simulate potentially infinite data streams from online process mining.

In this paper, we present the Distributed Event Factory, which fulfills the aforementioned requirements for distributed data generation. In Section 2, we present the basic concepts of the tool and show how it can be configured. In Section 3, using the example of a smart factory, we demonstrate that the Distributed Event Factory can generate semantically meaningful data. Section 4 discusses the maturity of the tool. Section 5 summarizes the work and outlines future work.

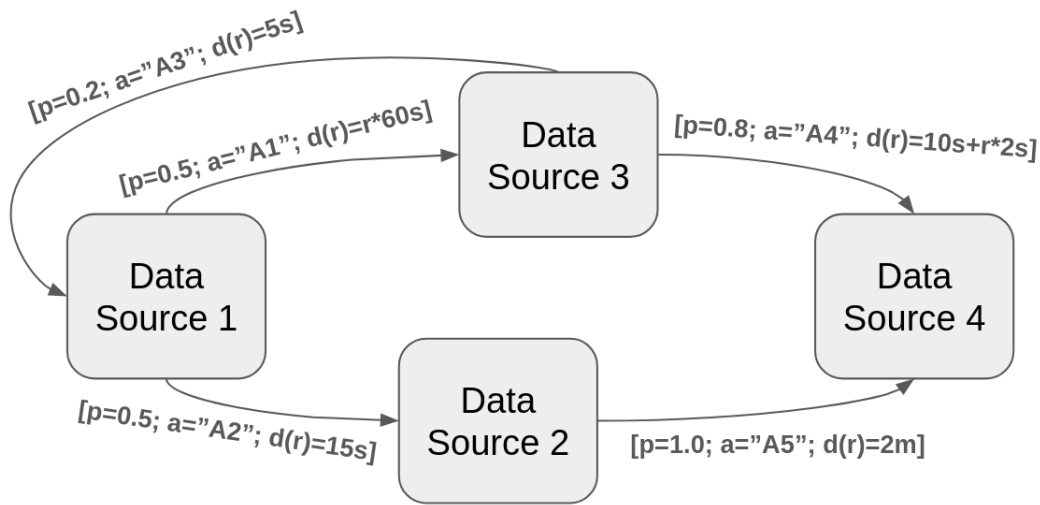
## 2. Distributed Event Factory

The Distributed Event Factory DEF is built upon a Markov chain. A Markov chain is a graph that models stochastic processes, where the probabilities of the next state transition depend only on the current state and not on previous states and is mostly used to simulate processes [7] and for load generation [8, 9]. The benefit of a Markov chain is the ability of simulation of real world settings as required for our purpose. DEF generates data according to the XES format [10] with an event defined by its case id, activity name, and a timestamp. Moreover, the event data source is added to the event. Formally, we describe the tool as follows:

**Definition 2.1** (Event). Let  $\mathcal{C}$  be the set of case ids,  $\mathcal{A}$  the set of all activities, and  $\mathcal{N}$  the set of data sources. We consider  $t \in \mathbb{N}$  as a timestamp, and define  $\mathcal{D} : () \rightarrow \mathbb{N}$  as the set of random functions that generate a process duration. The events are drawn from the set  $\mathcal{E} = \mathcal{C} \times \mathcal{A} \times \mathcal{T} \times \mathcal{N}$ .

**Definition 2.2** (Data Source Topology). Let the data source topology be represented as a graph  $G = (V, E)$ , where the vertices  $V$  correspond to data sources, each associated with a name  $n \in \mathcal{N}$ . Each edge in  $E$  has a probability inscription ( $p \in \mathbb{R}_{\leq 1}^+$ ) that refers to the data source where the process continues, a duration function  $d \in \mathcal{D}$ , and an activity  $a \in \mathcal{A}$ . Events are generated by a random instantiation of the graph. Transitions are activated due to previous edge inscription and the duration functions between vertices. The sum of the probabilities of all outgoing edges per node has to be 1.

**Definition 2.3** (Distributed Event Stream). Let  $S_n$  be the event stream of data source  $n \in \mathcal{N}$ . Furthermore,  $(a, d)$  are the activity and duration of the transition.  $t \in \mathbb{N}$  indicated the last tracked timestamp and  $c \in \mathcal{C}$  the current case id. Then the generated event is appended to the event stream of data source  $n$ :  $S_n \cdot \langle (c, a, t + d(), n) \rangle$



**Figure 1:** Exemplary Markov Chain for a Smart Factory process. Assuming the process (identifies by case id  $c=0$ ) is currently located at *Data Source 1* at timestamp  $t=0s$ . The process either continues with activity A1 or A2 with a probability of each 50%. We assume that the process continues with A2. Then, event  $e_1 = (c=0, a=A2, t=15s, n=DS1)$  is appended to the event log of Data Source 1.

In the following, we summarize how data distribution, varying sending rates, and the inclusion of temporal aspects have been implemented in the Distributed Event Factory.

**Data Distribution.** Every data source writes its own event stream. Hence, the data distribution is achieved. Thus, data is stored decentrally. Each data source transcripts the event of the outgoing edge to its event log. The tool provides three data sinks per default that emit events in real time: the console, a GUI, and a Kafka<sup>1</sup> broker. Additionally, individual data sinks can be defined that directly invoke a distributed process mining algorithm.

**Data Volume and Velocity.** The Distributed Event Factory can define arbitrary functions that define the speed at which the simulation should run. This allows control over the volume

<sup>1</sup><https://kafka.apache.org/>

and velocity of the data flow. A constant and a gradually increasing load has been implemented. However, these can also be overwritten arbitrarily.

**Process Execution Time.** Each edge in the Markov chain is assigned a stochastic function. This allows determining how long the process takes to execute. In practice, not every process has the same duration, and this can be modeled accordingly. It may occur that processes arrive at the processing components late or out of order. This can be simulated by allowing to assign a negative time duration. The Distributed Event Factory provides implements functions that model constant, uniformly distributed, or normally distributed execution times. These can also be configured and overwritten by the user accordingly.

The Distributed Event Factory is implemented in Python and publicly available on GitHub. It is configured using a YAML file.

### 3. Case Study: Smart Factory



**Figure 2:** The data source topology in the use case of a smart factory

**Figure 3:** Data source topology in the use case of a smart factory

Let us consider a warehouse and a factory. Both are run in different locations and follow particular processes (see Figure 4). (i) The activities of the warehouse include receiving goods, storage, picking, packing, shipping, managing inventory, and handling returns. (ii) The activities of the factory process are receiving goods, material preparation, assembly line setup, assembly, quality control, packaging, and shipping. Please note that the factory requires the material provided by the warehouse. Hence, the warehouse and the factory processes depend on each other. Any delays (e.g., inventory shortages, shipping delays), can significantly impact the factory's operations. Thus, an interorganizational simulation (i.e., over distributed locations) is required.

The locations in the Distributed Event Factory can be modeled as groups of data sources. Each group represents a specific operational area. For example, the process of receiving goods can be defined as a data source. The group id for this process would be "warehouse". In contrast, the assembly line setup would have "factory" as the group id.

Each data source can emit different values depending on its configuration. For instance, the goods reception produces activities like *Reject*, *Pass To Production*, or *Store*. These values allow for tailored responses based on specific operational needs.

Thus, by defining data sources and their corresponding group IDs, Distributed Event Factory can organize data sources by location.

```
type: load
loadType: gradual
tickCount: 100
load: 10
minimalLoad: 5
dataSourceTopology:
  defaultSink:
    type: console
  dataSources:
    - name: "GoodsDelivery"
      eventGeneration:
        selection: genericProbability
        distribution: [0.1, 0.9]
      from:
        events:
          - activity: "Reject"
            transition: 7
            duration: 1
          - activity: "Store"
            transition: 0
            duration:
              type: gaussian
              mu: 720
              sigma: 120
```

Figure 4: Example configuration of DEF via a yaml file.

## 4. Maturity

The Distributed Event Factory is currently a prototype stage. However, the requirements we set were implemented. Given that, sophisticated evaluations are still outstanding to validate the different configurations of DEF. In the future we plan to conduct these evaluations and also to advance DEF. We plan to implement a user interface, which initially allows visualization of the distributed process and we plan to extend it for interaction. Based on our testing DEF currently supports the generation of approximately 50000 simulation steps per second which makes it suitable for load testing. The number varies with the selected data sink. Currently, data is generated compliant to the XES format. An extension to object-centric event logs (OCEL) [11] or to IoT data formats such as NICE [12] is planned. Furthermore, concept drift could also be implemented.

## 5. Conclusion

This paper introduced the Distributed Event Factory, a tool addressing data generation for distributed process mining. The tool relies on Markov chains and incorporates a spatial component into process mining. It allows configuration of distribution data properties, supporting temporal dependencies and location of processes and its adjustment of data volume and velocity. Future work will focus on advancing the tool's maturity. Additionally, paradigms such as object-centric process mining and drifting data will be integrated. In this way, the Distributed Event Factory allows to evaluate different settings of data for distributed process mining.

## Acknowledgments

This work received funding by the Deutsche Forschungsgemeinschaft (DFG), grant 496119880.

## References

- [1] A. Burattin, Streaming process mining, in: *Process Mining Handbook*, Springer, 2022, pp. 349–372. doi:10.1007/978-3-031-08848-3\_11.
- [2] J. Andersen, P. Rathje, O. Landsiedel, Edgealpha: Distributed process discovery at the data sources, 2024. URL: <https://arxiv.org/abs/2405.03426>. arXiv:2405.03426.
- [3] K. Kaczmarek, A. Koschmider, Conceptualizing a log generator for privacy-aware event logs, *EMISA Forum* 41 (2021) 39–40.
- [4] J. Grimm, A. Kraus, H. van der Aa, CDLG: a tool for the generation of event logs with concept drifts, in: *International Conference on Business Process Management*, 2022. URL: [https://ceur-ws.org/Vol-3216/paper\\_241.pdf](https://ceur-ws.org/Vol-3216/paper_241.pdf).
- [5] Y. Zisgen, D. Janssen, A. Koschmider, *Generating Synthetic Sensor Event Logs for Process Mining*, Springer International Publishing, 2022, p. 130–137. doi:10.1007/978-3-031-07481-3\_15.
- [6] A. Burattin, Plg2: Multiperspective processes randomization and simulation for online and offline settings, 2015. doi:10.48550/ARXIV.1506.08415.
- [7] S. Chib, E. Greenberg, Markov chain Monte Carlo simulation methods in econometrics, *Econometric theory* 12 (1996) 409–431.
- [8] C. Vögele, A. van Hoorn, E. Schulz, W. Hasselbring, H. Krcmar, WESSBAS: extraction of probabilistic workload specifications for load testing and performance prediction—a model-driven approach for session-based application systems, *Software & Systems Modeling* 17 (2018) 443–477. doi:10.1007/s10270-016-0566-5.
- [9] A. van Hoorn, C. Vögele, E. Schulz, W. Hasselbring, H. Krcmar, Automatic extraction of probabilistic workload specifications for load testing session-based application systems, *EAI Endorsed Transactions on Self-Adaptive Systems* 15 (2015). doi:10.4108/icst.valuetools.2014.258171.
- [10] H. Verbeek, J. C. Buijs, B. F. Van Dongen, W. M. Van der Aalst, XES tools, in: *22nd International Conference on Advanced Information Systems Engineering (CAiSE 2010)*, CEUR-WS.org, 2010.
- [11] W. M. P. van der Aalst, Object-centric process mining: Dealing with divergence and convergence in event data, in: *Software Engineering and Formal Methods*, Springer, 2019, pp. 3–25. doi:10.1007/978-3-030-30446-1\_1.
- [12] Y. Bertrand, S. Veneruso, F. Leotta, M. Mecella, E. Serral, NICE: The Native IoT-Centric Event Log Model for Process Mining, in: *Process Mining Workshops*, Springer, 2024, pp. 32–44. doi:10.1007/978-3-031-56107-8\_3.