Software Impacts 17 (2023) 100551

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

Beamline: A comprehensive toolkit for research and development of streaming process mining

Andrea Burattin

Technical University of Denmark, Kgs. Lyngby, Denmark

ARTICLE INFO

Keywords:
Process mining
Streaming process mining
Apache Flink
Reactive programming

ABSTRACT

Beamline is a software library to support the research and development of streaming process mining algorithms. Specifically, it comprises a Java library, built on top of Apache Flink, which fosters high performance and deployment. The second component is a Python library (called pyBeamline, built using ReactiveX) which allows the quick prototyping and development of new streaming process mining algorithms. The two libraries share the same underlying data structures (`BEvent`) as well as the same fundamental principles, thus making the prototypes (built by researchers using pyBeamline) quickly transferrable to full-fledged and highly scalable applications (using Java Beamline).

Code metadata

Current code version	1.0.3
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2023-314
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/6952422/tree/v1
Legal Code License	Apache-2.0
Code versioning system used	git
Software code languages, tools, and services used	Python, Java
Compilation requirements, operating environments & dependencies	PM4py, reactivex
If available Link to developer documentation/manual	https://www.beamline.cloud
Support email for questions	andbur@dtu.dk

1. Introduction

Process mining [1,2] is an established field that seeks to integrate data science and process science to enhance processes and their executions. Process mining encompasses several sub-tasks, including "control-flow discovery" [1], which focuses on deriving a control-flow model based on the executions of the model itself. Another sub-task is "conformance checking" [3], which aims to validate that process executions align with a normative process description. In practical scenarios, control-flow discovery can be applied to gain insights into how a company manufactures or handles goods, with the aim of understanding the actual processes in order to optimize them. On the other hand, conformance checking finds applications in domains such as clinical protocols, ensuring that the observed processes align with the expected protocols. This enables the early detection of potential mistreatment of patients.

Traditional process mining techniques utilize *event log* files, which are structured as XML files following the IEEE XES standard [4]. These files contain events associated with a specific time period, and the process mining analyses are conducted within that timeframe. In streaming process mining, instead of working with static files, the input consists of an *event stream* [5]. Similar to event stream processing, streaming process mining aims to analyze data in real time and update the analysis promptly.

The field of streaming data analysis [6] imposes certain computational requirements that directly translate to the streaming process mining discipline [7]. Furthermore, in the latter, the need to establish conceptual connections between multiple data points observed at different timestamps (e.g., to be able to correlate events that belong to the same process instance) introduces complexity associated with the observation window (i.e., the time period during which the analysis takes place) [8].

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

E-mail address: andbur@dtu.dk.

https://doi.org/10.1016/j.simpa.2023.100551
Received 29 June 2023; Received in revised form 16 July 2023; Accepted 20 July 2023

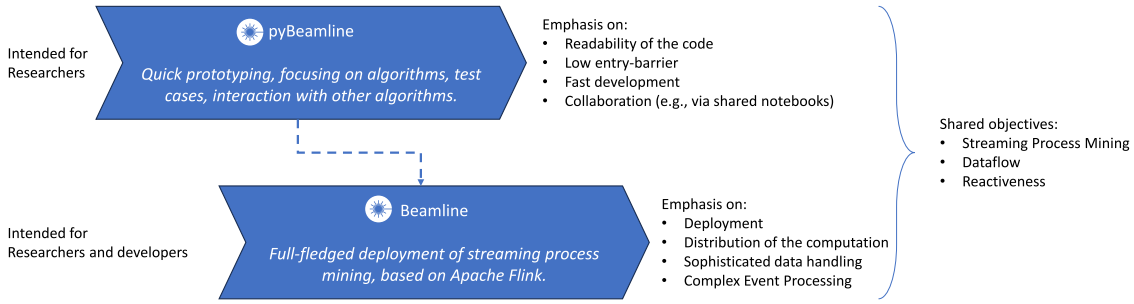

Fig. 1. General overview of Beamline and pyBeamline and the corresponding design principles. While pyBeamline is meant for researchers, Beamline is meant for developers as well, so it is expected that after an algorithm is prototyped and tested in pyBeamline, it will then be ported to Beamline.

This paper introduces Beamline: a comprehensive toolkit designed to support both researchers and developers in prototyping and deploying streaming process mining algorithms.

2. Description

Beamline comprises two libraries: the first one is built in Java (for simplicity we refer to this version just as Beamline), and the second is built in Python (called pyBeamline). The idea of the toolkit is that the two libraries serve as a comprehensive toolkit for researchers and developers of streaming process mining algorithms. While pyBeamline is meant for researchers (thus emphasizing collaboration, code readability, and fast development), the Java library is designed for both researchers and developers (thus emphasizing performance aspects). The two libraries share their general objectives, data structures, naming conventions, and programming principles and are devised to be used sequentially: first, an algorithm is prototyped by researchers on Python notebooks using the pyBeamline; and once maturity is reached, a complete implementation is ported to the Java library, as depicted Fig. 1.

While several non-functional requirements can be identified for streaming process mining algorithms (such as constant computational complexity in the processing of a single event), from a functional point of view, these algorithms are characterized by two fundamental aspects: reactiveness and their configuration via data flows:

- Reactiveness indicates that the algorithms should operate in an event-driven fashion in order to react to events in a responsive and efficient manner. In these approaches, the flow of data is represented as a stream of events that can be observed and reacted to.
- Data flows are graphs that represent the flow of data within a system: edges indicate the movement of data (in our case events) and nodes represent the processes or components that handle the data. An example of data flow could comprise connecting to an event source, filtering out certain events, sending them to a control-flow discovery algorithm, and finally storing the models in a time series database.

Both these requirements are fulfilled in both Beamline and pyBeamline.

2.1. pyBeamline (Python)

pyBeamline is a Python library that leverages the functionality of ReactiveX[1] and its Python counterpart, RxPY.[2] pyBeamline extends the capabilities of RxPY by providing process mining operators that facilitate the analysis of data streams in a reactive and event-based programming environment. This means that process mining-specific functionalities can be seamlessly integrated into the programming paradigm provided by RxPY.

[1] https://reactivex.io/

[2] https://rxpy.readthedocs.io/

2.2. Beamline (Java)

The Java library Beamline [9] is built on top of Apache Flink[3] [10], which is a library designed for distributed stateful computations over data streams. Apache Flink enables the creation of pipelines, that define the transformations to be applied to each event within the stream.

Beamline extends the capabilities of Apache Flink by introducing additional operations, particularly focused on process mining transformations. These transformations include process-aware event filters and flat-mappers, which are used for tasks such as process discovery or conformance computation. As Beamline leverages Apache Flink, all event transformations (both pre- and post-processing) and all the data connectors implemented are accessible.

3. Impact

Beamline and pyBeamline are currently being used for research and teaching purposes. The Java version of the library, which has been available since 2022, has been successfully used in several publications and teaching activities. pyBeamline is also currently being used as part of research and education activities.

The Java library currently implements control-flow discovery [11–15], conformance checking [16,17], and simulation [18] algorithms.[4] When comparing the impact of implementing streaming process mining algorithms using Beamline, as opposed to other process mining Java platforms, the benefit of the proposed library becomes clear: previous implementations, such as the Streaming Heuristics Miner's in ProM [19][5] required the specification of the entire event streaming/event handling infrastructure [20] as all open-source process mining frameworks assume that algorithms operate on static resources (e.g., a finite event log).

The Python library implements control-flow discovery [11,13] and conformance checking [16] algorithms. If we compare pyBeamline with the only other process mining Python library, PM4Py [21], we can appreciate that, despite a few streaming process mining algorithms being implemented in PM4Py, these have essentially no support for dataflow, nor for reactiveness or actual streams.

The other major process mining library, bupaR [22], designed for the R language, has no support for streaming whatsoever.

Table 1 reports a comparison of the 3 major open source process mining frameworks (ProM, PM4Py, and bupaR) with respect to the objectives set for Beamline. As can be noticed, while some goals can be partially achieved by other platforms, only Beamline has built-in support for a proper streaming infrastructure.

[3] https://flink.apache.org/

[4] A non-exhaustive list of techniques implemented in Beamline is available at https://www.beamline.cloud/implemented-techniques/.

[5] This implementation is available as open source at https://svn.win.tue.nl/repos/prom/Packages/StreamHeuristicsMiner/Trunk/.

Table 1
Comparison among the main open source process mining libraries with respect to the objectives of a streaming process mining framework.

Library	Streaming algorithms	Stream as input	Dataflow	Reactiveness
ProM [23]	✓	*Partial, ad hoc impl.*	×	×
PM4Py [21]	✓	×	×	×
bupaR [22]	×	×	*Partial*	×
Beamline *(this paper)*	✓	✓	✓	✓

Beamline is licensed according to Apache-2.0 terms.

4. Conclusion

As the field of process mining continues to advance, the ability to analyze data in real time and adapt to dynamic changes becomes increasingly crucial. Streaming process mining offers the potential to monitor and analyze processes as they unfold, enabling organizations to make timely interventions and optimizations. Indeed, a key requirement for advancing the field is the availability of a robust platform that enables researchers to rapidly develop new algorithms and allows developers to efficiently implement them with reliability and performance. This is precisely the objective that Beamline was designed to achieve. By providing a comprehensive and user-friendly platform, Beamline aims to lower the barriers to entry for both researchers and developers in the realm of streaming process mining. It offers a framework that facilitates the creation, implementation, and testing of new algorithms, thereby accelerating the innovation and adoption of novel techniques in this evolving field.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The author would like to thank Magnus Frederiksen who helped with porting some algorithms from Beamline to pyBeamline.

References

[1] W.M. van der Aalst, Process Mining, Springer, ISBN: 9783662498507, 2016, http://dx.doi.org/10.1007/978-3-662-49851-4.
[2] I.T.F. on Process Mining, Process Mining Manifesto, in: F. Daniel, K. Barkaoui, S. Dustdar (Eds.), Business Process Management Workshops, Springer-Verlag, 2011, pp. 169–194.
[3] J. Carmona, B. van Dongen, A. Solti, M. Weidlich, Conformance Checking, Springer International Publishing, 2018, http://dx.doi.org/10.1007/978-3-319-99414-7.
[4] C.W. Günther, E.H.M.W. Verbeek, XES Standard Definition, 2009, URL http://www.xes-standard.org/.
[5] A. Burattin, M. Eigenmann, R. Seiger, B. Weber, MQTT-XES: Real-time telemetry for process event data, in: CEUR Workshop Proceedings, 2020, URL http://ceur-ws.org/Vol-2673/.
[6] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: Massive Online Analysis Learning Examples, J. Mach. Learn. Res. 11 (2010) 1601–1604.
[7] A. Burattin, Streaming Process Discovery and Conformance Checking, in: S. Sakr, A.Y. Zomaya (Eds.), Encyclopedia of Big Data Technologies, Springer International Publishing, 2018, pp. 1–8, http://dx.doi.org/10.1007/978-3-319-63962-8_103-1.
[8] A. Burattin, Streaming Process Mining, in: W.M. van der Aalst, J. Carmona (Eds.), Process Mining Handbook, Springer, 2022, pp. 349–372, http://dx.doi.org/10.1007/978-3-031-08848-3_11.
[9] A. Burattin, Streaming process mining with beamline (extended abstract), in: CEUR Proceedings of the ICPM Demo Track 2022, CEUR-WS.org, 2022, pp. 75–79.
[10] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, K. Tzoumas, Apache Flink™: Stream and Batch Processing in a Single Engine, in: Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2015, pp. 28–38.
[11] A. Burattin, Process Mining Techniques in Business Environments, in: Lecture Notes in Business Information Processing, vol. 207, Springer International Publishing, Cham, ISBN: 978-3-319-17481-5, 2015, p. 220, http://dx.doi.org/10.1007/978-3-319-17482-2.
[12] A. Burattin, M. Cimitile, F.M. Maggi, A. Sperduti, Online Discovery of Declarative Process Models from Event Streams, IEEE Trans. Serv. Comput. (ISSN: 1939-1374) 8 (6) (2015) 833–846, http://dx.doi.org/10.1109/TSC.2015.2459703.
[13] A. Burattin, A. Sperduti, W.M. van der Aalst, Control-flow Discovery from Event Streams, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, ISBN: 9781479914883, 2014, pp. 2420–2427, http://dx.doi.org/10.1109/CEC.2014.6900341.
[14] A. Burattin, Streaming Process Discovery and Conformance Checking, in: S. Sakr, A. Zomaya (Eds.), Encyclopedia of Big Data Technologies, Springer International Publishing, 2018, http://dx.doi.org/10.1007/978-3-319-63962-8_103-1.
[15] A. Burattin, H.A. López, L. Starklit, Uncovering change: A streaming approach for declarative processes, in: M. Montali, A. Senderovich, M. Weidlich (Eds.), Process Mining Workshops, Springer Nature Switzerland, Cham, ISBN: 978-3-031-27815-0, 2023, pp. 158–170.
[16] A. Burattin, S.J. van Zelst, A. Armas-Cervantes, B.F. van Dongen, J. Carmona, Online conformance checking using behavioural patterns, in: M. Weske, M. Montali, I. Weber, J. vom Brocke (Eds.), Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings, in: Lecture Notes in Computer Science, 11080, Springer, 2018, pp. 250–267, http://dx.doi.org/10.1007/978-3-319-98648-7_15.
[17] A. Burattin, Online soft conformance checking: Any perspective can indicate deviations, 2022, arXiv:2201.09222.
[18] A. Burattin, PLG2 : Multiperspective Process Randomization with Online and Offline Simulations, in: Online Proceedings of the BPM Demo Track 2016, CEUR-WS.org, 2016.
[19] A. Burattin, A. Sperduti, W.M. van der Aalst, Heuristics Miners for Streaming Event Data, 2012, ArXiv CoRR URL http://arxiv.org/abs/1212.6383.
[20] S.J. van Zelst, A. Burattin, B. van Dongen, E.H.M.W. Verbeek, Data streams in ProM 6: A single-node architecture, in: CEUR Workshop Proceedings, vol. 1295, (ISSN: 16130073) 2014.
[21] A. Berti, S.J. van Zelst, W.M. van der Aalst, Process Mining for Python (PM4Py): Bridging the Gap between Process-and Data Science, in: Proc. of ICPM Demo Track, 2019.
[22] G. Janssenswillen, B. Depaire, M. Swennen, M. Jans, K. Vanhoof, bupaR: Enabling reproducible business process analysis, Knowl.-Based Syst. 163 (2019) 927–930.
[23] E.H.M.W. Verbeek, J. Buijs, B. van Dongen, W.M. van der Aalst, ProM 6: The Process Mining Toolkit, in: BPM 2010 Demo, 2010, pp. 34–39.