Streaming Process Discovery for Lambda Architecture-based Process Monitoring Platform

Anatoliy Batyuk
Dept. of Automated Control Systems Institute
of Computer Science and Information Technologies
Lviv Polytechnic National University
Lviv, Ukraine
abatyuk@gmail.com

Volodymyr Voityshyn
Dept. of Automated Control Systems Institute
of Computer Science and Information Technologies
Lviv Polytechnic National University
Lviv, Ukraine
voytyshyn@gmail.com

Abstract — **Having precise information about how business processes are performed in real-life is an important competitive advantage for a modern company. The digital footprint left by processes in IT systems can be transformed to event data for further analysis by process mining techniques. One of the biggest challenge of process mining is to deal with streaming event data and provide operational support for the on-going processes. Current paper is devoted to a streaming process discovery algorithm implemented by the authors upon a near real-time process monitoring platform which is based on the lambda architecture.**

Keywords* — *process mining, streaming process discovery, lambda architecture, Heuristic Miner, event data, event logs, XES, digital footprint

I. Introduction

Modern software systems are build with the purpose to automate various kind of processes. Some processes are well structured and implemented by means of workflow management systems using special languages like BPMN for process flow definition; however, most of the processes are executed under the circumstances when clear structuring is not possible. It takes place for example when flows of the running processes hardly depends on the end user's decisions. One of the main analysis task is to build process flows using real-life data generated by their execution. These tasks lay in the responsibility area of the discipline called process mining. This discipline is an emerging research and technological area. It acts as a connection that links such widely known domains as business process management (BPM), data science, and business intelligence (BI). From the high-level standpoint there are three categories of process mining tasks: (a) process discovery; (b) conformance checking; (c) model enhancement [1]. The goal of the first category is to create process model representation using digital footprint left by a process (this kind of data is called event data or event logs [1]). The tasks from the second category look for the difference between the real-life models discovered from event data and their "ideal" counterparts defined, for example, by means of BPMN or BPEL. And finally, the tasks for the third category make step further providing improvements to the existing process models.

During last decade the IT industry has shown significant changes towards data processing in near real-time mode with fast reactions on the happened events. This trend has influenced process mining as well. Most process discovery techniques were designed with the assumption that the data sets are static during analysis. However, necessity to work with so-called event data streams puts process mining into the position when it is needed from one hand to obtain data from event data streams and from the other hand to handle incomplete process instances. The process mining algorithms that deals with such kind of tasks are referred as streaming process discovery (SPD) [3, 4].

Current paper is devoted to implementation of a streaming process discovery technique within the scope of the near real-time process monitoring platform [5] based on lambda architecture [6]. The platform is designed to work with "small" data; however, its architecture design makes possible scaling to Big Data tasks. The implemented process discovery algorithms is based on the Heuristic Miner and consist of two parts: (a) batch processing and (b) data stream processing. The first part is based on conventional Heuristic Miner algorithm for offline process mining, [7] whilst the second one employs the Heuristic Miner modification for streaming event data [3].

The rest of the paper is structured as follows: brief description of the near real-time process monitoring platform is provided in section II; the task statement of the algorithm implementation is done in section III; section IV contains rationales behind the decision of choosing a process discovery technique; details about implementation of the algorithm is represented in section V; section VI provides results of algorithm's execution on a sample event data set; concluding remarks are in section VII.

II. Process Monitoring Platform

The purpose of the process monitoring platform [5] is to provide operational support for on-going processes in near real-time mode. The platform is an extensible solution that provides built-in functions (e.g. control flow visualization, prediction of a process instance completion time [8], automatic suggestion of next steps [9], alerting if an anomaly is detected) with capability to integrate custom features specific to a practical application. Architecture concept of the platform is depicted on Fig. 1.

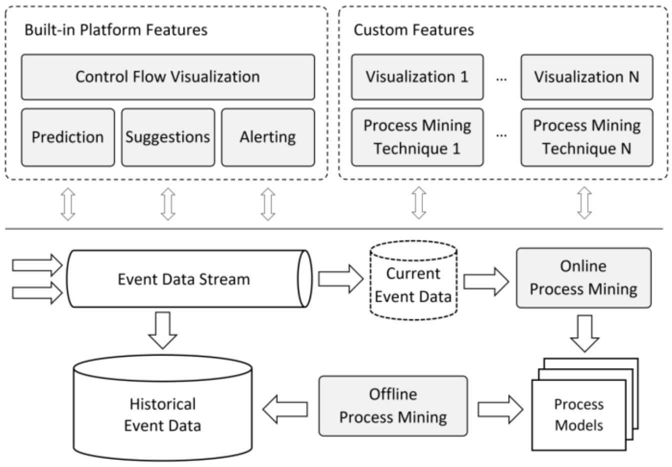

Fig. 1. Architecture concept of the process monitoring platform

Event data is received as a stream through a message queue. It is distinguished event data related to completed (historical event data persisted in a database) and incomplete process instances (current event data which is cached in memory).

Design of the platform is based on the idea of the lambda architecture pattern [6] so that it is included two types of data processing mechanisms: (a) offline and (b) online. The algorithms from the first category are executed against the entire data set persisted in the database; whilst the online algorithms are intended to process event data once it is obtained from the data stream. A similar data stream processing solution is represented in [10].

Current implementation of the platform targets "small" data; however, its architecture design allows scaling to a big data solution like the system from [11].

III. Task Statement

As defined above the represented process mining technique includes two types of algorithms: (a) offline and (b) online. To maintain compatibility with industry standards the algorithms receive event data in XES format [12]. Each event data item contains the following required attributes: case, activity, and timestamp.

The offline process mining algorithm takes as an input historical event data persisted in the database. The output of this algorithm is a control flow model. The model is displayed by means of the built-in control flow visualization feature.

A sequence of event data items obtained from the stream is an input for the online process mining algorithm. Number of event data items included into the sequence is equal or more than 1. The online algorithm produces updates for the process model generated by the offline algorithm. The updates are visualized in near real-time mode.

It is accepted the following limitations of the implemented process mining technique:

- It targets so-called stationary data streams; in other words, handling concept drift [13] of a process model is not supported in the current version.
- Processes with many kinds of events and transitions among them (so-called "spaghetti" processes) are not currently supported.

Addressing the limitations above is planned (section VII).

IV. Choosing a Process Discovery Technique

Despite of the fact that the process mining is a relatively new research and engineering field a few process discovery techniques have already been created and examined against practical tasks with real-life event data. For the purposes of the current task a process discovery algorithm has to fulfill the following requirements:

- The output model is readable to for the end users who are domain experts but not necessarily experts in process mining.
- Offline and online process discovery are supported.

One of the very first and simplest process discovery technique is the alpha algorithm [14]. This algorithm takes event data and produces a Petri net. One of the biggest disadvantage of it is that in case of real-life even data the mined models are hardly readable for the end user because the algorithm shows all the transitions without consideration of their importance.

More applicable for real-life event data is the Fuzzy Miner [15]. Its purpose is to deal with so-called "spaghetti" processes that is processes with many events and complicated transitions among them. The basic idea behind this approach is to represent a complex process on a certain level of abstraction. The level is configured by means of the set of predefined metrics. This algorithm has proved its efficiency in practice and has been adopted by Disco [16] and Celonis [17].

Heuristic Miner [7] is another process discovery algorithm aimed to deal with real-life data. Like the Fuzzy Miner it takes into account frequencies of activities and transitions so that infrequent paths are not incorporated in the produced model.

Additionally, pragmatic comparison of process discovery algorithms is provided in [18].

According to the task statement (section III) the most suitable in current case is the Heuristic Miner. The idea of its

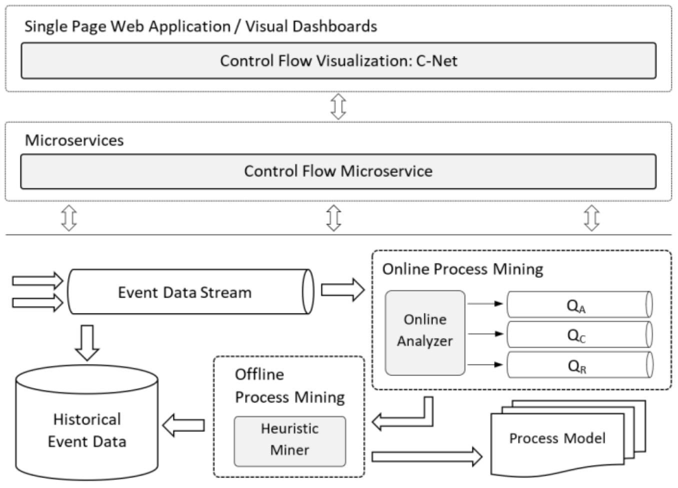

Fig. 2. Streaming process discovery for the process monitoring platform

streaming implementation within the scope of the process monitoring platform is based on the Heuristic Miner for stationary data streams [3].

V. Implementation of the Streaming Process Discovery Algorithm

The implemented algorithm is based on the Heuristic Miner for event data streams [3]. Current version is suitable for stationary data streams and does not support evolving streams.

Mapping of the algorithm's implementation to the platform's architecture is depicted on Fig. 2. The main points of the implementation are the following:

- Event data is received from the Event Data Stream in XES format [12]. The received event data is persisted in the Historical Event Data storage and also processed by the Online Analyzer.
- The Online Analyzer maintains the following 3 queues: (a) Q_A, the most recently received activities with weights of their importance; (b) Q_C, the most recent activities for each case; (c) Q_R, the most recent direct transitions with weights of their importance. The queues are implemented by means of an in-memory database.
- The Online Analyzer triggers the batch Heuristic Miner algorithm when it is necessary to update process model.
- As the historical event data is accumulated all the time, the batch Heuristic Miner algorithm applies the sliding window approach to limit a data set used for mining.
- An updated process model is passed to the Control Flow Visualization by means of the Web Sockets [19] technology so that the end user does not need to reload the page manually to see the updates.

The server-side implementation is mostly Java based, and the frontend part uses HTML5, in particular, process control-flows are visualized by means of SVG [20] and D3.js [21].

VI. Calculation Results

To test the implemented streaming process discovery technique the Road Traffic Fine Management Process [22] dataset has been used. Metrics of the dataset are represented in Table 1.

TABLE I. Metrics of the Dataset

Metric	Value
Number of processes	1
Number of process instances	150370
Number of events	561470
Number of event classes	11
Start date	01 Jan 2000
End date	18 Jun 2013

Using the dataset a data stream has been simulated. The implemented streaming process discovery algorithm has generated a control-flow model in near real-time mode. Two sample snapshots of the generated dynamic model are represented on Fig. 3 and Fig. 4 (the figures have been redrawn manually, because the visual models generated by the platform and not compact enough to be put into a publication).

To compare the results produced by the implemented platform the Flexible Heuristic Miner [23] plug-in of ProM 6.7 has been executed against the entire dataset. The result generated by ProM (Fig. 5) matches the outcomes of the platform.

VII. Concluding Remarks

The implemented streaming process discovery technique for the lambda architecture-based platform has demonstrated satisfactory results on test datasets. The produced models match the models generated by the Flexible Heuristic Miner plug-in on the entire data sets. Further tasks are to deal with concept and handle so-called “spaghetti” processes.

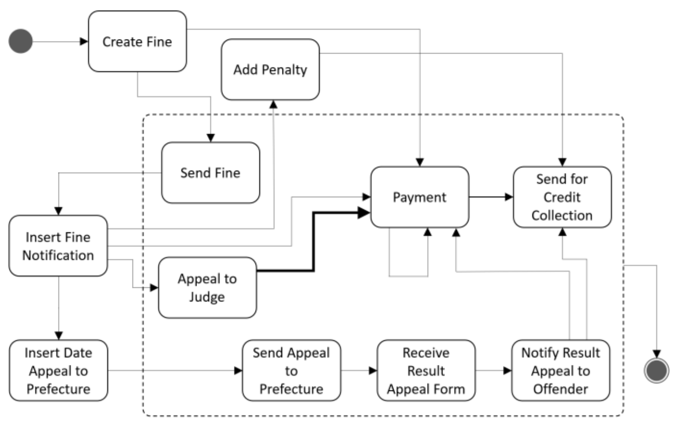
Create Fine
Add Penalty
Send Fine
Payment
Send for Credit Collection
Insert Fine Notification
Appeal to Judge
Insert Date Appeal to Prefecture
Send Appeal to Prefecture
Receive Result Appeal Form
Notify Result Appeal to Offender

Fig. 3. Control-flow model of the Road Traffic Fine Management Process dataset for the 2000-2005 years timeframe

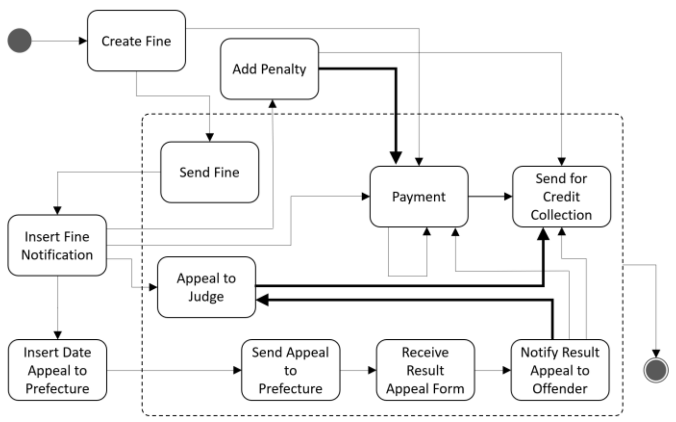
Create Fine
Add Penalty
Send Fine
Payment
Send for Credit Collection
Insert Fine Notification
Appeal to Judge
Insert Date Appeal to Prefecture
Send Appeal to Prefecture
Receive Result Appeal Form
Notify Result Appeal to Offender

Fig. 4. Control-flow model of the Road Traffic Fine Management Process dataset for the 2008-2013 years timeframe

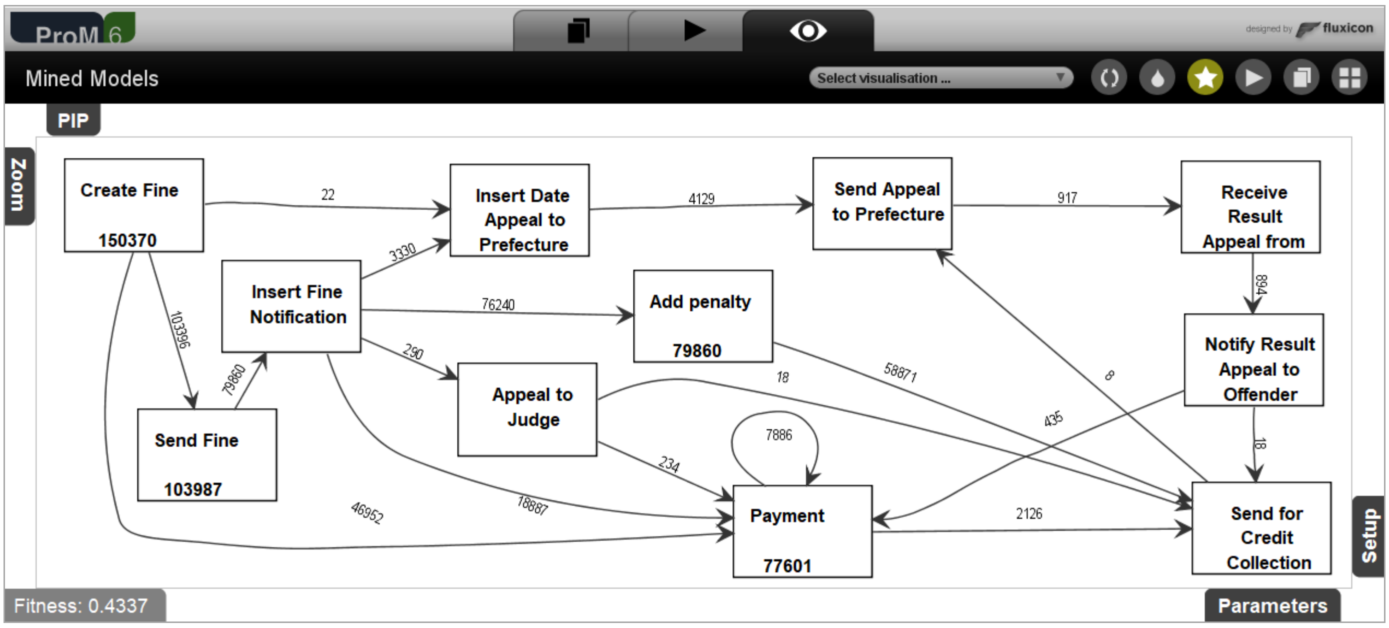

Fig. 5. Control-flow model of the Road Traffic Fine Management Process dataset produced by the Flexible Heuristics Miner plug-in (ProM 6.7)

References

[1] W.M.P. van der Aalst, et al., "Process mining manifesto," in Business Process Management Workshops. BPM 2011 International Workshops. Lecture Notes in Business Information Processing, Clermont-Ferrand, France, 2011, vol. 99, pp. 169-194.

[2] W.M.P. van der Aalst, Process mining: data science in action, 2nd ed. Berlin: Springer, 2016.

[3] A. Burattin, A. Sperduti, W.M.P. van der Aalst, "Heuristics Miners for Streaming Event Data," in CoRR (Computing Research Repository), abs/1212.6383, 2012.

[4] A. Burattin, "Process Mining for Stream Data Sources,". in Process Mining Techniques in Business Environments. Lecture Notes in Business Information Processing, Cham, Springer, 2015, vol 207, pp. 177-204.

[5] A. Batyuk and V. Voityshyn, "Software Architecture Design of the Real-Time Processes Monitoring Platform," in 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP 2018), Lviv, Ukraine, 2018, unpublished.

[6] "Lambda Architecture", [Online]. Available: http://lambda-architecture .net/. [Accessed: 25 Mar 2018].

[7] A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves De Medeiros, "Process Mining with the Heuristics Miner Algorithm," Eindhoven: BETA Working Paper Series, WP 166, 2006.

[8] O. Mulesa, F. Geche, A. Batyuk, V. Buchok, "Development of Combined Information Technology for Time Series Prediction," in Advances in Intelligent Systems and Computing II (CSIT 2017), Lviv, Ukraine, 2017, vol 689, pp. 361-373.

[9] O. Mulesa, F. Geche, V. Voloshchuk, V. Buchok and A. Batyuk, "Information technology for time series forecasting with considering fuzzy expert evaluations," 2017 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv, 2017, pp. 105-108.

[10] I. Tsmots, O. Skorokhoda, T. Tesliuk and V. Rabyk, "Designing features of hardware and software tools for intelligent processing of intensive data streams," 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP), Lviv, 2016, pp. 332-335.

[11] A. Batyuk and V. Voityshyn, "Apache storm based on topology for real-time processing of streaming data from social networks," in 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP 2016), Lviv, Ukraine, 2016, pp. 345-349.

[12] IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams, IEEE Std 1849-2016, 2016.

[13] R. P. Jagadeesh Chandra BoseWil, M. P. van der Aalst, I. Žliobaitė, M. Pechenizkiy, "Handling Concept Drift in Process Mining," in Advanced Information Systems Engineering. CAiSE 2011. Lecture Notes in Computer Science, London, UK, 2011, vol 6741, pp. 391-405.

[14] W.M.P. van der Aalst, T. Weijters, L. Maruster, "Workflow mining: discovering process models from event logs," in IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 9, pp. 1128-1142, 2004.

[15] W.G. Christian, W.M.P. van der Aalst, "Fuzzy Mining – Adaptive Process Simplification," in Proceedings of the 5th International Conference on Business Process Management (BPM 2007), Brisbane, Australia, 2007, vol 4714, pp. 328-343.

[16] Ch.W. Günther, A. Rozinat, "Disco: Discover Your Processes," in Proceedings of the Demonstration Track of the 10th International Conference on Business Process Management (BPM 2012), Tallinn, Estonia, 2012, vol 940, pp. 40-44.

[17] F. Veit, J. Geyer-Klingeberg, J. Madrzak, M. Haug, J. Thomson, "The Proactive Insights Engine: Process Mining meets Machine Learning and Artificial Intelligence," in 15th International Conference on Business Process Management (BPM 2017). BPM Demo Track and BPM Dissertation Award, Barcelona, Spain, 2017, vol 1920.

[18] A. Rozinat, "ProM Tips - Which Mining Algorithm Should You Use?", 2010. [Online]. Available: https://fluxicon.com/blog/2010/10/prom-tips-mining-algorithm/. [Accessed: 25 Mar 2018].

[19] "Web sockets", [Online]. Available: https://html.spec.whatwg.org/ multipage/web-sockets.html#network. [Accessed: 14 Apr 2018].

[20] "Scalable Vector Graphics (SVG) 1.1 (Second Edition)", [Online]. Available: https://www.w3.org/TR/2011/REC-SVG11-20110816/. [Accessed: 14 Apr 2018].

[21] "D3", [Online]. Available: https://d3js.org/. [Accessed: 14 Apr 2018].

[22] "Road Traffic Fine Management Process", [Online]. Available: https://data.4tu.nl/repository/uuid:270fd440-1057-4fb9-89a9-b699b47990f5. [Accessed: 14 Apr 2018].

[23] A. J. M. M. Weijters, J. T. S. Ribeiro, "Flexible Heuristics Miner (FHM)," 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Paris, 2011, pp. 310-317.