https://doi.org/10.23939/sisn2019.01.070

UDC 004.42+519.85+004.9

DISTRIBUTED SOFTWARE SYSTEM WITH WEB INTERFACE FOR AUTOMATED BUSINESS PROCESS DISCOVERY

A. Ye. Batyuk[1], V. V. Voityshyn[2]
Lviv Polytechnic National University,
Institute of Computer Sciences and Information Technologies, ACS Department,
12, S. Bandery Str., Lviv, Ukraine
[1]Email: abatyuk@gmail.com, ORCID: 0000-0001-7650-7383
[2]Email: voytyshyn@gmail.com, ORCID: 0000-0002-7889-2593

As an applied discipline, process mining emerged about two decades ago, and its methods have been increasingly used in practice for recent few years. What differentiates process mining from the conventional data mining is considering process nature of the analyzed data. Rapid development of the process mining software market niche has risen relevance of such task as scalability of process mining methods. Adaptation of the process discovery method, called Fuzzy Miner, to distributed software systems with web interface has been proposed by the authors. To address the scalability requirements, the calculation procedures are implemented on different part of the system: the most computer resource consuming algorithms are executed on the server side whilst less resource consuming calculations are placed on the client side. In turns, the server-side components belong either to the data layer or service layer. The data layer is accountable for storing event data in XES format and measuring process metrics. Building a process graph and communication with the client web application is the responsibility of the service layer. The purpose of the client-side web application is to render a process graph generated in the server-side. The calculation logic is covered with unit and integration tests so that its correctness is checked automatically. In order to reduce total cost of ownership of the system, it is implemented with free software. From the performed calculations and comparison of the outcomes with the results received by means of the existing ProM 6.8 plugins (Fuzzy Miner and Alpha++ Miner), it can be concluded that the proposed adaptation of the Fuzzy Miner method ensures representation of the behavior seen in an event log (like the ProM 6.8 plugins successfully do). In turns, from the software architecture standpoint, the proposed solution demonstrates better scalability characteristics, i.e., ability to increase capacity in order to handle bigger amount of event data, in comparison with the mentioned above ProM plugins.

Keywords - process mining; event data; event logs; XES; ProM; Fuzzy Miner.

УДК 004.42+519.85+004.9

РОЗПОДІЛЕНА ПРОГРАМНА СИСТЕМА З WEB-ІНТЕРФЕЙСОМ ДЛЯ АВТОМАТИЗОВАНОЇ ПОБУДОВИ МОДЕЛІ БІЗНЕС-ПРОЦЕСУ

А. Є. Батюк[1], В. В. Войтишин[2]
Національний університет “Львівська політехніка”,
інститут комп’ютерних наук та інформаційних технологій, кафедра АСУ,
Україна, Львів, вул. С. Бандери, 12
[1]Email: abatyuk@gmail.com, ORCID: 0000-0001-7650-7383
[2]Email: voytyshyn@gmail.com, ORCID: 0000-0002-7889-2593

Як прикладна дисципліна, процес-майнинг виник два десятиліття тому, і упродовж останніх кількох років його методи отримують все ширше застосування на

практиці. Ознакою, яка відрізняє процес-майнинг від класичного дата-майнигу, є фокусування на процесній природі оброблюваних даних. Активний розвиток ніші ринку програмного забезпечення процес-майнингу загострив актуальність задач, пов'язаних із підвищенням масштабованості його методів. Автори пропонують подальший розвиток методу Fuzzy Miner, що розширює його застосування до розподілених програмних системах з web-інтерфейсом. Для забезпечення вимог масштабованості розрахункові процедури реалізовано у різних частинах системи: найбільш ресурсоємні алгоритми виконуються на стороні сервера, тоді як менш ресурсоємні – на стороні web-клієнта. Своєю чергою, серверні компоненти належать до одного із логічних рівнів: даних або сервісів. Рівень даних відповідає за збереження даних у XES-форматі та збирання процесних метрик. Побудова графу процесу та взаємодія з клієнтською частиною забезпечується рівнем сервісів. Призначенням клієнтської web-програми є відбраження графу процесу, який будує сервер. Автоматична перевірка коректності реалізації розрахункової логіки забезпечується модульними та інтеграційними тестами. Рішення, реалізоване авторами, ґрунтується на відкритих програмних продуктах, що дає змогу знизити сумарну вартість програмної системи. На основі порівняння отриманих значень із результатами, одержаними за допомогою ProM 6.8 (було застосовано плагіни: Fuzzy Miner and Alpha++ Miner), зроблено висновок, що запропонована адаптація методу Fuzzy Miner забезпечує відображення поведінки, яка присутня в event-даних (що також успішно виконують відповідні ProM-плагіни). Своєю чергою, з погляду системної архітектури запропоноване рішення демонструє кращі характеристики масштабованості, тобто здатності нарощувати обчислювальні потужності з метою опрацювання більших обсягів event-даних порівняно із згаданими вище ProM-плагінами.

Ключові слова: процес-майнинг; event-дані; логи; XES; ProM; Fuzzy Miner.

Introduction

The purpose of IT system of an organization is to automate operations which are done in order achieve the organization's goals. If such operations are repeated many times and even become standardized, they are usually called business processes. In current paper, the "business process" term is understood in the same meaning as in (Burattin, 2015). So, the main task of an IT system is to serve business needs of an organization by automating various kind of business process. In some cases, such business processes are well structured and predefined by means of special technologies, for example, business process management systems. However, in most cases the processes do not have predefined models; therefore, the sequence of activities is described as textual instructions or even shared among the participants verbally. One of the first task, which appears during analysis of those processes, is to visualize their flow chart. In the process mining field (van der Aalst et al, 2012), this task was named process discovery.

During the last two decades, it was developed various process discovery methods. One of them is the Fuzzy Miner (Günther & van der Aalst, 2007). What differentiates Fuzzy Miner from the other methods is the ability to deal with unstructured processes with many events and transitions providing the possibility to simplify their representation for the end users employing the so-called road-map concept. One of the earliest versions of Fuzzy Miner was implemented as a ProM plugin ("Fuzzy Miner", 2009). Since ProM (Verbeek, Buijs, van Dongen, & van der Aalst, 2010) is a desktop tool mostly used by scientific researchers, a major drawback of that Fuzzy Miner implementation is its limited availability for the end users who do not have academic background in process mining. Another disadvantage of the existing implementation is its insufficient scalability, i.e., practical impossibility to deal with increasing amount of event data.

Current paper is devoted to further development of the Fuzzy Miner. The adaptation of the method proposed by the authors addresses the mentioned above limitations of the existing implementations. The applied approach allows to improve scalability of the software system since significant part of the calculations can be done on the server side (which is not only much more powerful than the end users' machines but also allows to increase capacity by applying various scalability techniques (van Steen & Tanenbaum, 2017)). Additionally, it was introduced some

improvements of the process flow graph visualization, for example, it was added the start and end events (which are not displayed by the Fuzzy Miner ProM plugin).

The rest of the paper is divided into the following sections: problem statement; brief overview of the related works; description of the proposed method adaptation; software implementation of the method; calculations with comparison the results; and concluding remarks.

Problem Statement

The task is to design and implement an adaptation of the Fuzzy Miner to a distributed software system with web interface. Under the process discovery task, it is understood the general process discovery problem stated in (van der Aalst, 2016), i.e., finding a method that maps event log to a process flow graph so that the graph represents behavior seen in the event log. An important requirement, that differentiates the proposed adaptation of Fuzzy Miner from its predecessors, is scalability, i.e., ability to increase capacity in order to handle bigger amount of event data. It should be noted that the term "distributed software system" is used in the meaning provided in (van Steen & Tanenbaum, 2017), i.e., it is a software system which components are deployed on different network nodes and communicates with one another by sending messages. In case of a distribute system with web interface, its components belong to either server-side or client-side components.

Related Works

Fuzzy Miner was chosen as a basic process discovery technique for the solution represented in current paper. The main concepts of Fuzzy Miner can be found in (Günther & van der Aalst, 2007). One of the earliest software implementations of Fuzzy Miner was a ProM plugin ("Fuzzy Miner", 2009). Then, Fuzzy Miner was successfully adopted by commercial products like Disco (Günther & Rozinat, 2012). Comprehensive overviews of existing process mining algorithms (van der Aalst, 2016; Turner, Tiwari, Olaiya, & Xu, 2012) and also a pragmatic comparison of process discovery techniques (Rozinat, 2010) shows that Fuzzy Miner is one of the methods that match requirements of processing real-life event logs. Relevance of the task to design and implement a process discovery method for a distributed software system is proved in the recent process mining software overviews (van der Aalst, 2016; Batyuk & Voityshyn, 2018a).

Process Discovery Method

The key point of adapting the Fuzzy Miner to a distributed software system with web interface is to decide which steps are performed on the server side, and which of them are executed on the client side. The implemented process discovery method includes the following three steps (Fig. 1): (1) measuring process metrics; (2) building the process flow graph; (3) visualization of the graph.

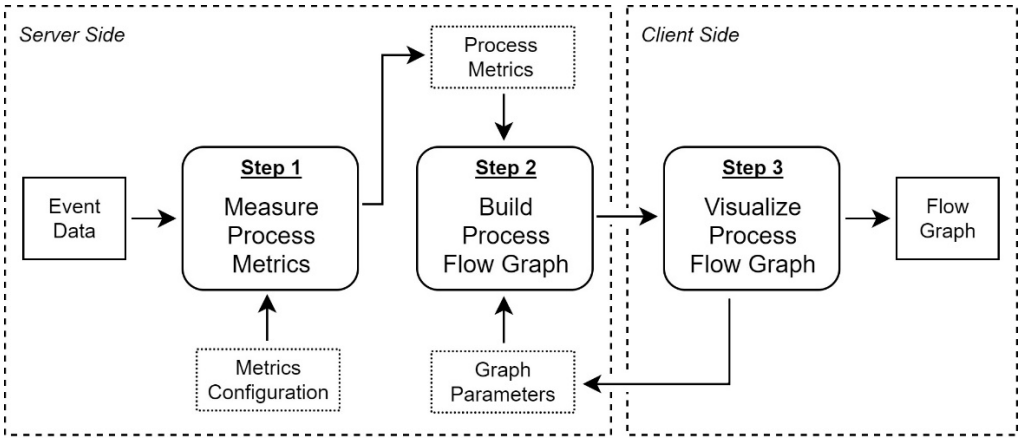

Fig. 1. Steps of the process discovery method

On step 1, the process metrics (Günther & van der Aalst, 2007; "Fuzzy Miner", 2009) are measured from the event data set. The measurement procedure is done according to set of configuration parameters

like the described in (“Fuzzy Miner”, 2009). Then, the collected metrics are used to build the process flow graph on step 2. It should be noted that parameters necessary to build the graph are passed from the client side (which allows to configure the graph according the end user’s needs). On the final step, the process flow graph is visualized on the client web page.

Software Implementation of the Method

From the architecture perspective, the implemented software system consists of the three layers: (1) data, (2) services, and (3) presentation. The components from the first two layers work on the server side, the ones from the third layer belong to the client side. The components and connectors model (Bass, Clements, & Kazman, 2012) of the software system is depicted in Fig. 2.

Event data is received from the external sources in XES format (IEEE Std 1849-2016, 2016) and then persisted in the event data storage. In the simplest case the event data storage component can be a text file (or a collection of text files) with XES-formatted data. In case of necessity to store big amount of event data, a more advanced solution, like scalable non-relational database (e.g. Apache Cassandra), can be implemented. From the stored event data, process metrics are calculated by a job implemented according to the batch processing pattern. The process metrics job uses configuration parameters like the ones from (“Fuzzy Miner”, 2009). The measured process metrics are stored in a non-relational storage: the simplest approach can employ JSON-formatted text files and more advanced implementations can be based on a NoSQL database, for example, MongoDB. It should be emphasized that one of the scalability options for the data layer is to execute multiple process metrics jobs simultaneously.

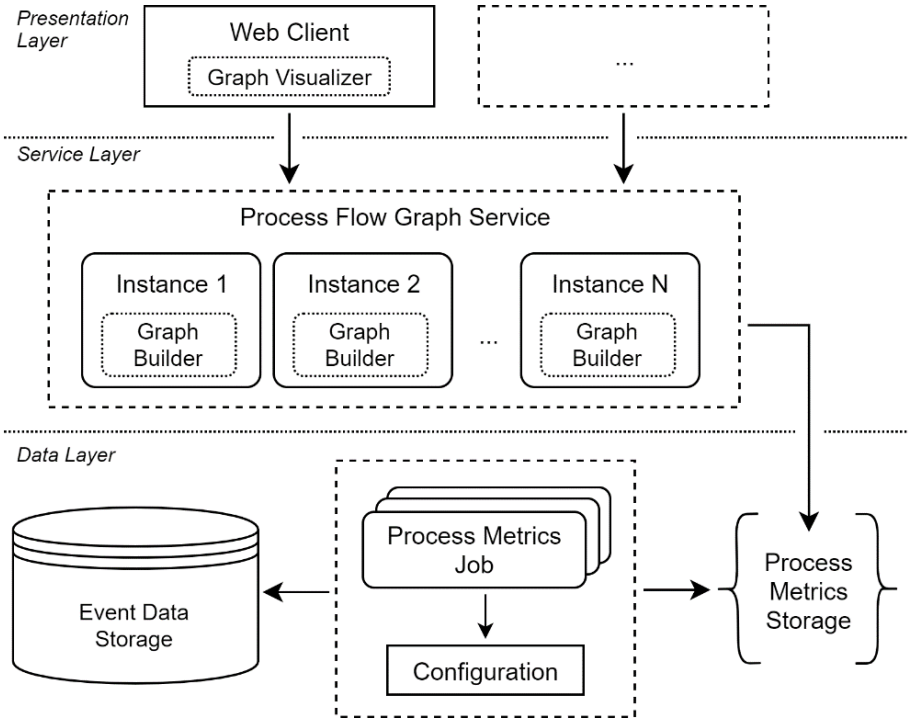

Fig. 2. Components and connectors model of the software implementation

Service layer includes a scalable RESTful service with the main responsibility to build a process flow graph using the process metrics collected on the data layer. In comparison with the data layer, the service layer requires less computational resources since it does not need to process event data items and uses as the input process metrics measured on the data layer. In turns, the main responsibility of the presentation layer is to visualize the process flow graph.

The technologies used by the authors to implement the described above software system are listed in Table 1. All the chosen technologies are open source (except MongoDB) and free to use for non-commercial purposes.

Table 1

Technologies used for the software implementation

Component	Technology	Version	Official web site
Web Client	Angular	7	https://angular.io
Graph Visualizer	d3js	5.9.0	https://d3js.org
Process Flow Graph Service, Graph Builder	Java / Open JDK	12	http://openjdk.java.net
	Spring Boot	2.1.3	http://spring.io
Event Data Storage	Apache Cassandra	3.11.4	http://cassandra.apache.org
Process Metrics Job	Java / Open JDK	12	https://mongodb.com
	OpenXES	-	http://www.xes-standard.org/openxes/start
Process Metrics Storage	MongoDB Community Server	4.0.8	https://www.mongodb.com

It should be noted that regardless the introduced scalability enhancements, the solution designed for the data layer can be further improved by adding streaming processing for the newly received event data in order to avoid repeating batch processing of the entire data set by the process metrics job.

Calculations and Comparing the Results

With the purpose to test the implemented process discovery method, the Road traffic fine management process public data set was used (de Leoni & Mannhardt, 2015). Characteristics of the chosen data set are represented in Table 2.

Table 2

Characteristics of the Road traffic fine management process data set

Metric	Value
Number of processes	1
Number of process instances	150370
Number of events	561470
Number of event classes (the start and end events are not included)	11
Start date	01 Jan 2000
End date	18 Jun 2013

The test was performed with the configuration listed below. On the data layer the following metrics were collected: (1) frequency significance (unary), (2) routing significance, (3) frequency significance (binary), (4) distance significance, (5) proximity correlation, (6) endpoint correlation, (7) originator correlation, (8) data type correlation, (9) data value correlation. For each metric, the Weight attribute was set up with "1.0" value, and the Invert attribute had "false" value; the n^{th}-root attenuation factor with power 2.7 was chosen. Process graph parameters passed from the client side to the server side with a request to build a process graph are listed in Table 3. The more details related to the applied configuration can be found in (Günther & van der Aalst, 2007; "Fuzzy Miner", 2009).

Table 3

Process graph parameters

#	Filter	Parameter	Value
1	Node filter	Significance cut-off	0.416
2	Edge filter	Best edges / Fuzzy edges	Fuzzy edges
3		Significance / Correlation ratio	0.75
4		Edges cut-off	0.20
5		Ignore self-loops	Yes
6		Interpret absolute	No
7	Concurrency filter	Filter concurrency	Yes
8		Preserve	0.60
9		Balance	0.70

Process graph built by the software implemented by the authors with the specified above configuration is depicted in Fig. 3. Short explanation to the used notation: (1) the process start event is represented with a completely painted circle; (2) the process end event is displayed as a painted circle with a white area insight; (3) the numbers in the left top corner are unique identifiers of the activities.

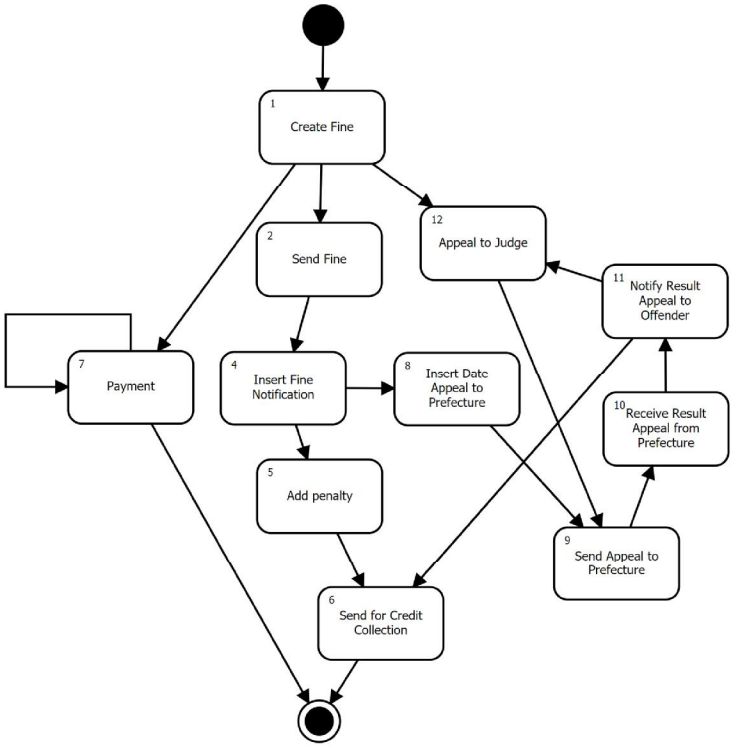

Fig. 3. Process graph visualized using the proposed software implementation

To compare result of the performed test (Fig. 3) with existing process discovery methods, process flow graph was built for the same data set using the Fuzzy Miner ("Fuzzy Miner", 2009) and Alpha++ Miner (Wen, van der Aalst, Wang, & Sun, 2007) ProM 6.8 plugins. The received results are represented in Fig. 4 and Fig. 5 respectively.

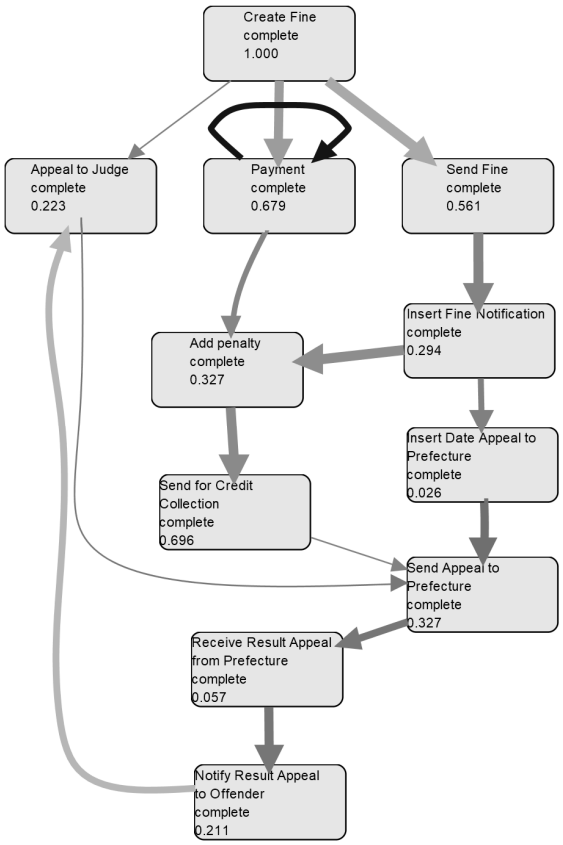

Fig. 4. Process graph built using the Fuzzy Miner (ProM 6.8)

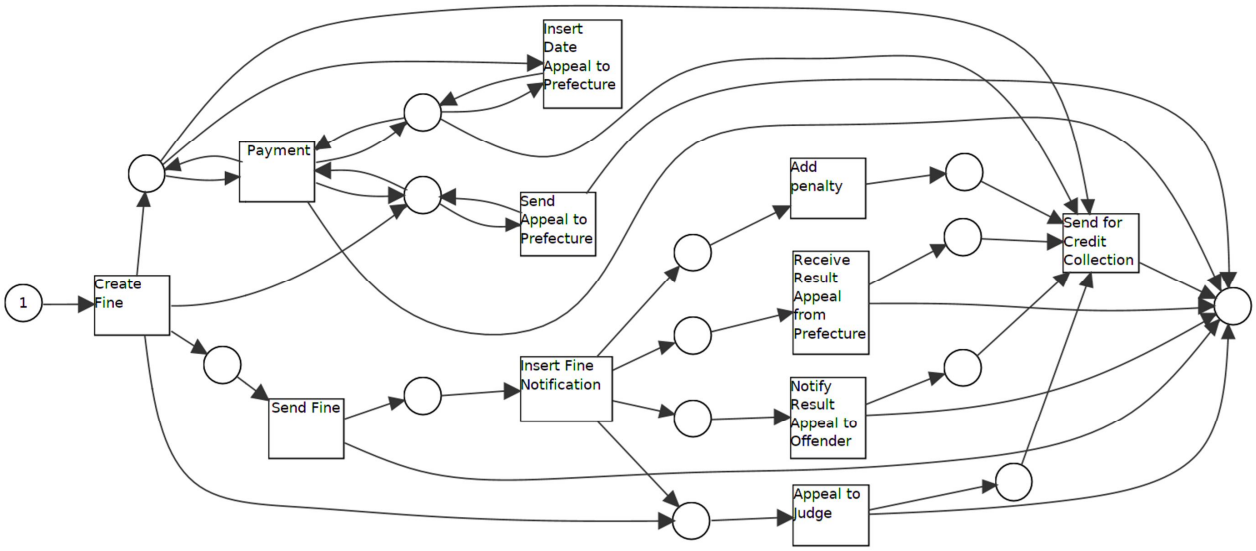

Fig. 5. Process graph built using the Alpha++ Miner (ProM 6.8)

From visualization standpoint, the advantage of the process graph from Fig. 3, in comparison with the one generated by means of Fuzzy Miner (Fig. 4), is the ability to display the start and end events. It should be emphasized that from Fig. 4 it is not clear where the process starts and ends. Comparing the graphs from Fig. 3 and Fig. 5, it can be concluded that the results received using the Alpha++ Miner is more difficult to read (especially for the end users without academic background in process mining). On one hand, the difficulty is caused by the Petri-net-based notation, which is completely correct from theoretical standpoint but is not easy to understand for the end users. And, on the other hand, the graph from Fig. 5 contains all the transition found in the event log, whilst the proposed implementation (like Fuzzy Miner) displays only the most significant ones hiding those which are not so important according to the collected process metrics. Also, it worth noting that it would be valuable for the graph from Fig. 3 to highlight importance for the displayed transitions, like the Fuzzy Miner does.

Concluding Remarks

The method proposed in current paper is further development of the Fuzzy Miner with the purpose to adapt it to distributed software systems with web interface. The enhancements introduced by the authors allowed to improve scalability of the solution from the software architecture standpoint, in comparison with the Fuzzy Miner ProM plugin.

From the business application point of view, the method can be an extension to a BI (business intelligence) solution used within an organization's IT system. What differentiate such extension from a conventional BI platform is the ability to visualize data, taking into account its process nature. Another example of possible application of the described method can be the batch data processor for the real-time business process monitoring platform (RTBMP) represented in (Batyuk & Voityshyn, 2018b).

The software implementation is based on open source software products (an exception is MongoDB) which allow to reduce total cost of ownership of the system. The designed architecture and chosen technologies address the scalability requirements by separating calculations between different components so that the most expensive calculations (collecting process metrics and building a process graph) are done on the server side. Relevant server-side components can be clustered depending on amount of processed event data and number of the end users.

It should be noted that the proposed solution has some limitations related to the taken approach for handling event data: (1) necessity to scan the entire data set during each execution of the process metrics job; (2) inability to react to recently received event data immediately. In case of applying the method for implementing the batch data processor of RTBPM (Batyuk & Voityshyn, 2018b) or incorporating the proposed method to the energy efficiency management system (Teslyuk, Tsmots, Teslyuk, Medykovskyy, & Opotyak, 2017), it is worth to overcome the 1st of the known disadvantages allowing to reduce necessary

computational resources and load on the event data storage. In current implementation it is assumed that all the process instances in the event log have finished. However, one of the future improvements can be the possibility to handle incomplete process instances (i.e. instances which are being executed at the moment when the process metrics job runs). Forecasting of process events (including the end event) based on the methods from (Mulesa, Geche, Batyuk, Buchok, 2018) is foreseen as a core of such improvement.

References

1. Burattin, A. (2015). Process Mining Techniques in Business Environments. Cham: Springer. doi: 10.1007/978-3-319-17482-2

2. van der Aalst, W.M. P., et al. (2012). Process mining manifesto. In F. Daniel, K. Barkaoui, S. Dustdar (Ed.), Business Process Management Workshops. BPM 2011 International Workshops. Lecture Notes in Business Information Processing, vol. 99 (pp. 169-194) Berlin, Heidelberg: Springer. doi: 10.1007/978-3-642-28108-2_19

3. Günther, Ch.W. & van der Aalst, W.M.P. (2007). Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics. In G. Alonso, P. Dadam, M. Rosemann (Ed.) Proceedings of the 5th International Conference on Business Process Management. BPM 2007. Lecture Notes in Computer Science, vol. 4714 (pp. 328–343) Berlin, Heidelberg: Springer. doi: 10.1007/978-3-540-75183-0_24

4. Fuzzy Miner. (2009, June 17). Retrieved from http://www.processmining.org/online/fuzzyminer

5. Verbeek, H.M.W., Buijs J.C.A.M., van Dongen B.F., & van der Aalst, W.M.P. (2010). ProM 6: the process mining toolkit. In M. La Rosa (Ed.), Proceedings of the Business Process Management 2010 Demonstration Track, vol. 615 (pp. 34–39). CEUR-WS.org.

6. van Steen, M. & Tanenbaum, A.S. (2017). Distributed Systems, 3rd ed., distributed-systems.net.

7. van der Aalst, W.M.P. (2016). Process mining: data science in action, 2nd ed. Berlin: Springer. doi: 10.1007/978-3-662-49851-4

8. Turner, C.J., Tiwari, A., Olaiya, R., & Xu, Y. (2012). Business Process Mining: From Theory to Practice, Business Process Management Journal, 18(3), pp. 493–512. doi: 10.1108/14637151211232669

9. Rozinat, A. (2010, Oct 18). ProM Tips – Which Mining Algorithm Should You Use?, Fluxicon. Retrieved from https://fluxicon.com/blog/2010/10/prom-tips-mining-algorithm/

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

11. Batyuk, A. & Voityshyn, V. (2018). Process Mining: Applied Discipline and Software Implementations, KPI Science News, 5, pp. 22–36. doi: 10.20535/1810-0546.2018.5.146178

12. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice, 3rd ed. Addison-Wesley Professional.

13. IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams. (2016). IEEE Std 1849–2016.

14. OpenXES. (2017, June 16). Retrieved from http://www.xes-standard.org/openxes/start

15. de Leoni, M. & Mannhardt, F. (2015, Feb 13). Road Traffic Fine Management Process. Retrieved from https://data.4tu.nl/repository/uuid:270fd440-1057-4fb9-89a9-b699b47990f5

16. Wen, L., van der Aalst, W.M.P., Wang, J., & Sun, J. (2007). Mining process models with non-free-choice constructs, Data Mining and Knowledge Discovery, 15(2), pp. 145–180.

17. Batyuk, A. & Voityshyn, V. (2018). Software Architecture Design of the Information Technology for Real-Time Business Process Monitoring, ECONTECHMOD, 7(3), pp. 13–22.

18. Teslyuk, T., Tsmots, I., Teslyuk, V., Medykovskyy, M., & Opotyak, Y. (2017). Architecture of the management system of energy efficiency of technological processes at the enterprise. 2017 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT) (pp. 429-433). Lviv. doi: 10.1109/STC-CSIT.2017.8098822

19. Mulesa, O., Geche, F., Batyuk, A., Buchok, V. (2018). Development of Combined Information Technology for Time Series Prediction. In Shakhovska, N., Stepashko, V. (Ed.), Advances in Intelligent Systems and Computing II. CSIT 2017. Advances in Intelligent Systems and Computing, vol. 689. Cham: Springer. doi: 10.1007/978-3-319-70581-1_26