ECONTECHMOD. AN INTERNATIONAL QUARTERLY JOURNAL – 2018, Vol. 07, No. 3, 13 – 22

Software architecture design of the information technology for real-time business process monitoring

Anatoliy Batyuk and Volodymyr Voityshyn

Automated Control Systems Department
Lviv Polytechnic National University
Lviv, Ukraine
e-mail: abatyuk@gmail.com, voytyshyn@gmail.com

Received August 26.2018: accepted October 20.2018

Abstract. Having precise understanding of how business processes are performed in real-life is an important input for decision makers and consequently is a strong competitive advantage for an organization. In the constantly changing modern business environment it is crucial to provide that information as soon as possible, preferably in the real-time mode. In practice, such kind of tasks are usually resolved by means of Business Intelligence solutions implemented either from scratch or based upon customizable packages. Despite of the wide range currently available types of data visualizations, modern BI solutions still lacks features to represent data obtained from process-aware systems, for example control flow charts. Current paper is devoted to the information technology for real-time business process monitoring. The represented solution is an extendable software which is based on the lambda architecture and a streaming process discovery technique.

Keywords: information technology, process mining, streaming process discovery, lambda architecture, Heuristic Miner

INTRODUCTION

The main purpose of modern enterprise IT systems is to ensure automation of various kind of business processes. Sometimes processes are well structured and implemented by means of workflow management systems using specialized languages (like BPMN) for process flow definition; however, in most cases processes are executed under the circumstances when clear definition of their flows is quite difficult or even not practically possible. For example, this takes place when a process is implemented by means of a few separate integrated software systems, and flow of the process is very dependent on people's decisions. This usually leads to quite complicated transitions among process steps with many branches and parallel executions. Nonetheless, monitoring of such kind of business processes using their "digital footprints" (i.e. real-life data generated by processes execution) is an important practical task which ensure representation of valuable insights for decision makers of how business operations are performed in real-life. Another challenge of modern enterprise software development is to supply necessary information as soon as possible so that the latest changes in business environment are considered, and relevant decisions are made by responsible people in time. Such kind of tasks are usually resolved by means of Business Intelligence solutions implemented either from scratch or upon customizable packages (e.g. Microsoft Power BI). Despite of the wide range of supported approaches to analyze and visualize data, in most cases those solutions do not consider process origination of the data, for example do not provide features of process flow charts visualization.

Current paper is devoted to software architecture design of the implemented by the authors real-time business process monitoring (shortly RTBPM) information technology. The represented solution ensures visibility and traceability of the executed business operations displaying them in the process-aware manner so that the end users can use the provided insights to make appropriate decisions in time. From technical implementation perspective the information technology is based on the lambda architecture [26]. The algorithmic part is built upon a streaming process discovery technique [11]. It is necessary to note that in the context of current paper "real-time" means a period of time (starting from occurring an event and its processing by means of RTBPM) during which the response of the information technology is relevant. An exact value of that period of time is not specified because in different practical applications the value can variate from a few seconds to even tens of minutes (further information about performance requirements can be found in section 3.2).

It is worth to notice that current paper focuses on the real-time process monitoring functionality. So, the features which are typical for such kind of software products (e.g. security, logging, user management etc.) but not related to the main task, are out of scope of the article.

The rest of the paper is structured as follows: the problem is stated in section 2; technical task (including functional and non-functional requirements) is defined in section 3; section 4 is devoted to technical architecture of RTBPM; the process flow discovery method is described in section 5; concluding remarks are in the final section.

CONTEXT AND PROBLEM STATEMENT

RTBPM can be considered as a specialized type of a business intelligence (BI) solution that fulfills a gap which is typical for BI products – lack of process analytics features. Before starting describing the information technology itself it is important to outline in what context it is intended to be used.

As a rule, a corporate IT ecosystem includes many separate software applications, for example, a customer relationship management, document management, human resource management, enterprise resource planning, and many others. Each of those systems is not by itself but is integrated with the rest of an ecosystem (discussing of enterprise software integration approaches is out of scope). Another common characteristic is that almost each system has a database (or even a few databases). To provide information about actual situation within the organization data from those applications is consolidated into a so-called data warehouse (for example, by means of extract-transform-load or shortly ETL tools) and then consumed by a reporting tool which, in turns, visualizes the data in necessary perspectives (Fig. 1). It should be noted that the model on Fig. 1 is a simplified because in real-life a corporate IT ecosystem may include more than one data warehouse as well as master data management, business process management, enterprise service bus etc.

A corporate IT ecosystem is usually aimed to automate certain amount of business processes. In a most common scenario those processes are implemented without explicit flow definition, and their execution involve more than one software product. An issue with that is lack of clear, precise, and up to date information of how those business processes are performed in real-life. Such kind of process analysis usually starts (but is not limited to) with the following tasks: (a) process flow visualization; (b) collecting metrics to identify weaknesses and bottlenecks; (c) comparison of the actual and pre-defined process models. The expected outcomes of the analysis are usually recommendations of how to improve or even redesign the existing business processes.

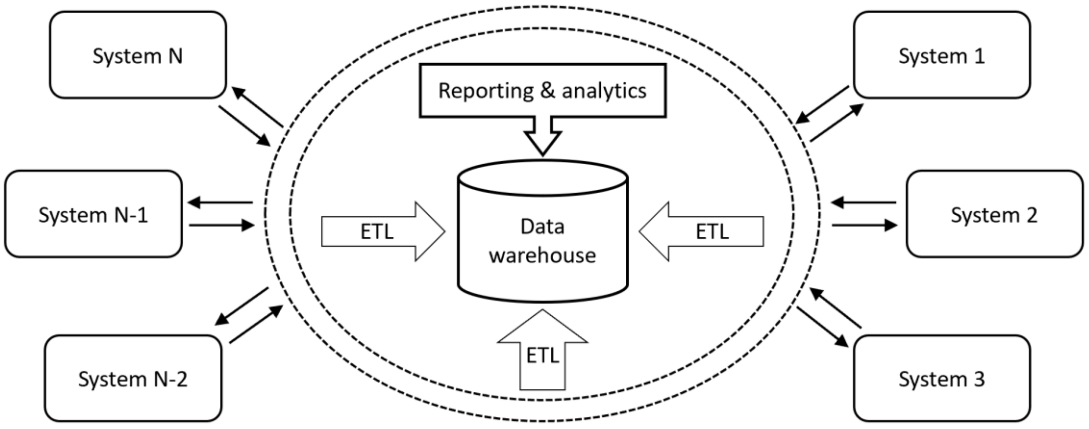

Fig. 1. Model of a corporate software ecosystem from data warehousing and reporting perspective

Typical approach to deal with that task is to hire experts who perform audit and provide suggestions. As during the audit phase inputs are collected and analyzed by the involved experts mostly manually, this way has significant drawbacks such as time consuming, low precision of analysis results, strong dependency from the expert's perception. Another important disadvantage is that collected inputs and outcomes quickly become outdated due to rapid changes in the business environment. So, in case of necessity to update previously obtained results, the audit has to be performed one more time almost from the very beginning. However, it should be noted that most part of necessary data is already accumulated by software systems. And the question is how to reduce manual efforts by automating mining that data and turning it into valuable insights. Proposed by the authors solution of the stated problem is represented in the further sections.

Technical task

On the high level the task is to implement an information technology with the purpose to address the problem of real-time business process monitoring. Statement of the technical task is decomposed into the following three parts: (a) functional requirements, (b) non-functional requirements, and (c) requirements to the process discovery method.

Functional requirements

The core feature of RTBPM is to visualize a process flow updating it in real-time mode with accordance to receiving events from a continuous data stream. It should be noted that a predecessor of the current solution is described in [6]. Next features are based on the ideas of the refined process mining framework [1]: (a) prediction of a process instance completion time (based on the approach form [18]); (b) suggestions of next steps for the end users; and (c) alerting if actual state of a process breaches predefined rules. Like the process flow visualization functionality, the specified analytics features should work in real-time mode.

Non-functional requirements

Obviously, the list of tasks related to real-time processes monitoring is not limited by the functions specified above. Those four features are included into a so-called functional core; however, the designed software architecture should be flexible enough so that it can be adapted to a specific practical application. The used software architecture building approach is based on the attribute-driven design method [4] which prescribes to specify so-called non-functional requirements as a subset of the quality attributes [5].

A. Performance. The main requirement to performance is that the system should process event data in real-time mode. In other words, such characteristics as latency (the interval from the time of receiving of an event till the time when the end user sees the changes caused by the event) and throughput (number of events processed by the system during a certain period of time) are highly important. Currently it is not possible to provide certain numeric thresholds for these characteristics since they depend on a hardware and software deployment infrastructure as well as number of running instances of the system's components (see the scalability requirements below).

B. Scalability in current context means the ability to variate latency and throughput according to the number of received events. First of all, it is important to decide what amount of data the system is intended to deal with. The reason behind necessity to make such decision is that a Big Data solution is usually more expensive in implementation and requires more powerful deployment infrastructure. However, in practice there are many cases when amount of processed data does not require Big Data technologies. That is why it has been decided that the represented version of RTBPM aims to deal with so-called "small" data (not Big Data). The difference between "small" and "big" data is explained in terms of so-called "5Vs" [12].

C. Interoperability for RTBPM has two aspects. From one hand, the software should fulfil so-called cloud agnostic requirements so that it can be deployed either on an organization's on-premises infrastructure or on a cloud computing platform (e.g. Microsoft Azure, Amazon Web Services, Google Cloud Platform etc.). From the other hand, the system should support the following types of integrations with the rest of a corporate IT ecosystem: (a) input and (b) output. The purpose of input integrations is to receive event data from other software systems for its further processing and storing in the event data storage. The output integrations are done by exposing API so that other software can consume data or listen for events generated by RTBPM (e.g. subscribing for alerts thrown when predefined rules are breached during execution of a process).

D. Extensibility. The represented version of RTBPM includes only core functionality and should allow to add new plug-ins necessary to a specific practical application (e.g. an anomaly detection module based on machine learning capabilities with appropriate visualization and alerting features).

E. Configurability in the context of current system is about ability of adaptation to specific application needs without changing the source code. For example, alerting rules are defined by the administrator of the system considering requirements to a specific process.

Requirements to the process discovery method

The primary requirement to the used process discovery method is the ability to handle received data in real-time mode. In the process mining field such a task is named "streaming process discovery" (SPD). The algorithms should take as the input a special type of data called "event data" (formal definition of this term is provided in [1]). To maintain industry standards event data should be received in XES format [25].

The following limitations are acceptable for the process discovery method implemented in current version of RTBPM: (a) no possibility to handle concept drift [15] of a process model; (b) visualization of so-called "spaghetti" processes [13] is not supported.

SOLUTION ARCHITECTURE

Integration with a corporate IT ecosystem

Before describing solution architecture of RTBPM it is important to outline what place it takes in an organization's IT. Considering the model from Fig. 1 there are two main approaches of RTBPM integration with the rest of an IT ecosystem. The first and simpler approach is to connect RTBPM with a single system (Fig. 2). In this case RTBPM listens for events which happen within the connected system; additionally, it is possible a reverse integration, for example, when the connected system receives alerts thrown by RTBPM. Technically all these integrations are supposed to be done with open cross-platform protocols (e.g. HTTP).

The second integration approach implies that RTBPM takes the central place next to a data warehouse and reporting tool (Fig. 3) so that it receives event data from the entire ecosystem. The integration API exposed by RTBPM can be consumed, for example, by the reporting tool. It also should be noted that conceptually current case is close to a business activity monitoring (BAM) [24]..

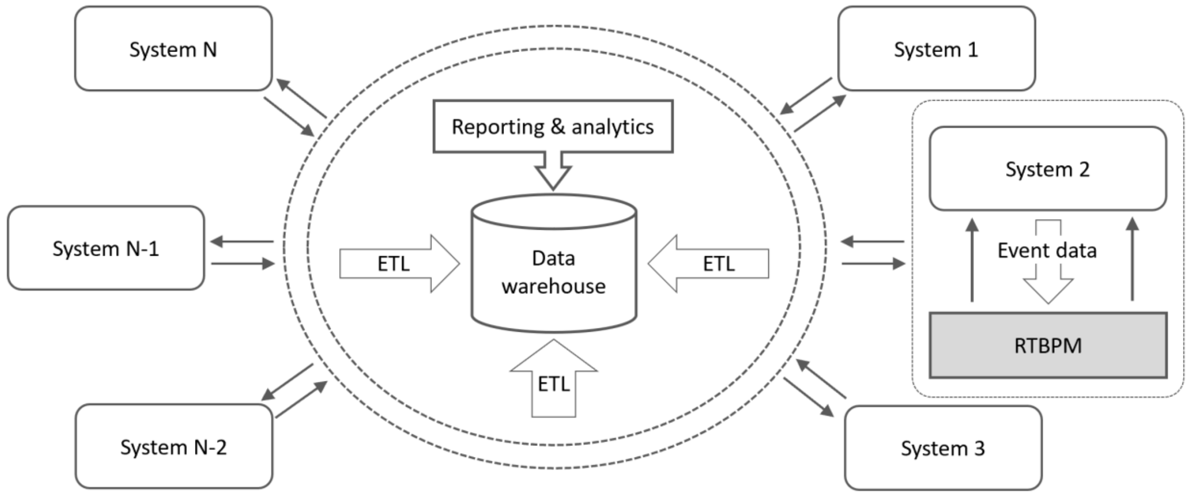

Fig. 2. Integration of RTBPM with a corporate software ecosystem: approach 1

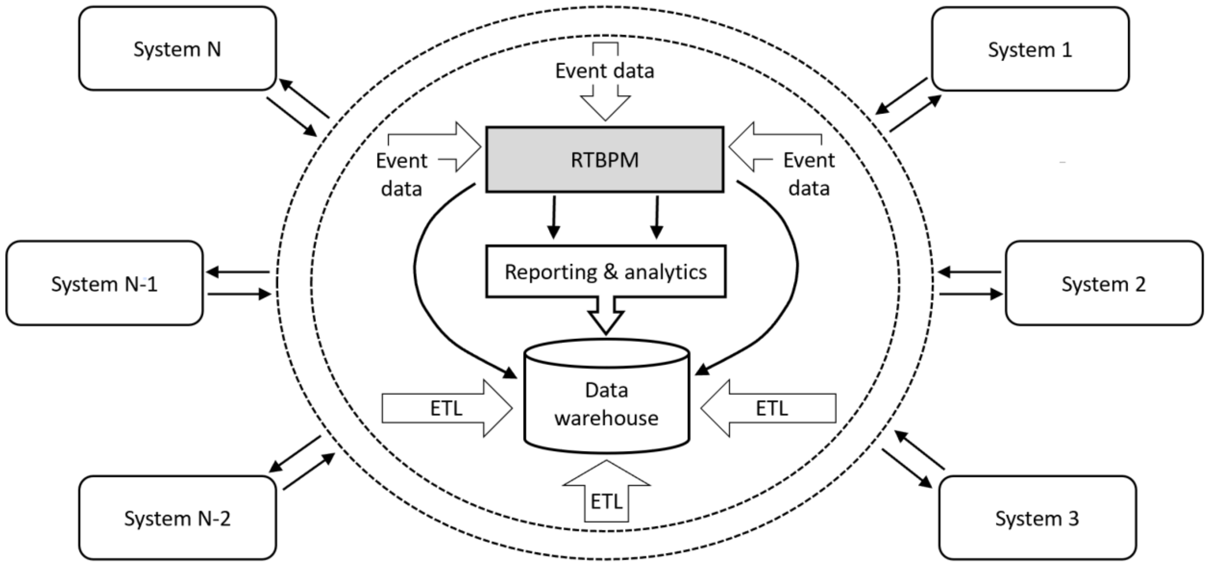

Fig. 3. Integration of RTBPM with a corporate software ecosystem: approach 2

RTBPM is intended to process event data related to business operation (e.g. creating an order, approving an invoice etc.) as well as technical messages handled by corporate middleware (e.g. an enterprise services bus, messaging queue etc.). It also should be noted that RTBPM integration capabilities allow to cover the three levels of process mining defined by Gartner [16]: (a) single process, (b) operational, and (c) organizational end-to-end level. The first integration approach matches (a) and partially (b); the second one potentially is able to ensure all the three levels.

Components model

The architecture of RTBPM (Fig. 4) is designed according to the functional and non-functional requirements defined in section 3. On the high level the system consists of the following logical parts: (a) presentation layer, (b) integration API, and (c) data processing layer.

The presentation layer consists of the frontend components as well as microservices with the purpose to serve the frontend. This layer is implemented in the pluggable manner so that the extensibility non-functional requirement is satisfied (see section 3.2). The integration API is consumed by external systems so that the data accumulated within the system and results of data processing are accessible to the surrounding corporate ecosystem (see the interoperability requirement in section 3.2).

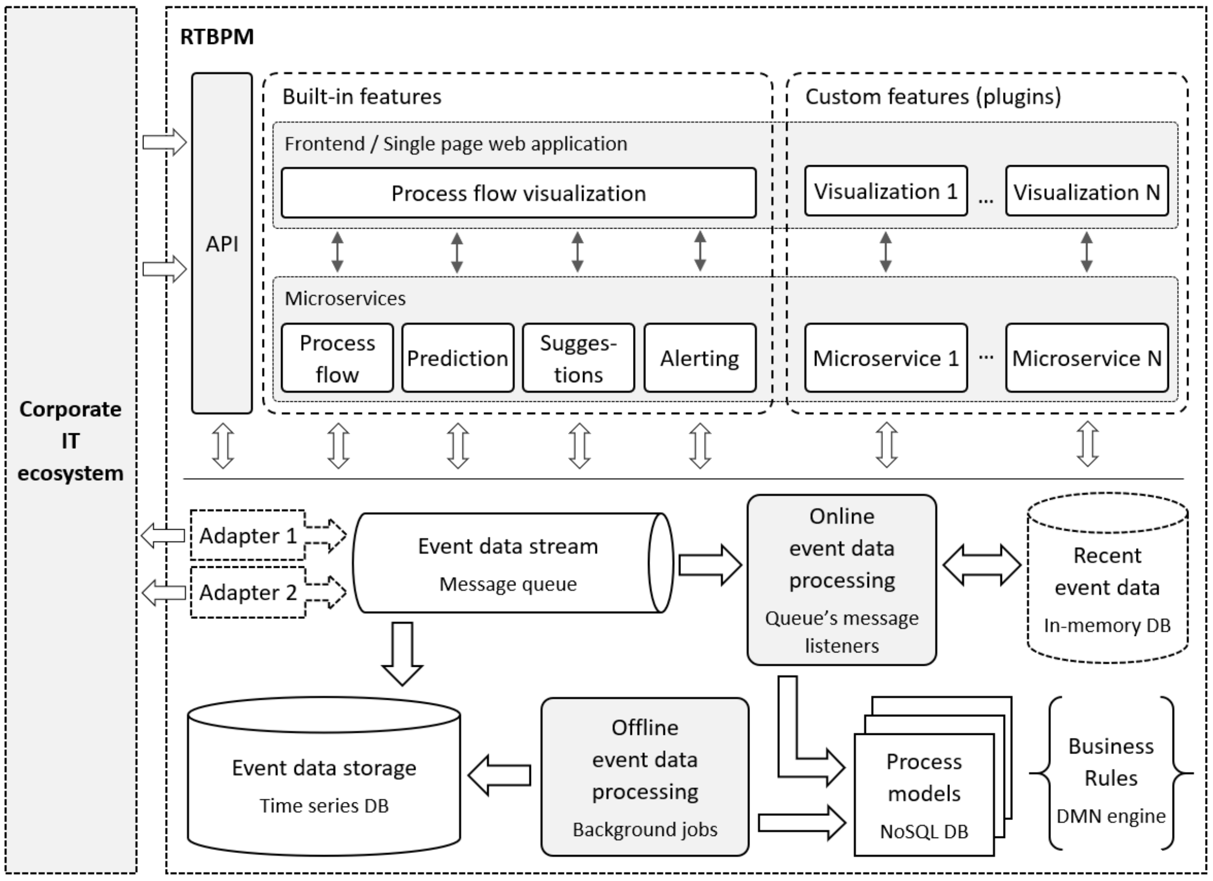

Fig. 2. Component model of RTBPM

The most important requirement is real-time event data processing. To meet such a requirement one of the following patterns are usually used: (a) lambda architecture [26] and (b) kappa architecture [19] (both have come from the Big Data field). The main peculiarity of the lambda architecture is that the data processing procedures are grouped in batch and speed layers. The first group is accountable for handling the entire data set including recent and historical data; consequently, those procedures are executed periodically and cannot handle data in real-time. In turns, the speed layer deals with an input stream ensuring data processing in real-time mode. The kappa architecture differs from the lambda architecture in that it does not include the batch layer. The implemented data processing methods (see more details in section 5) require both batch and speed layers. That is the reason of choosing the lambda architecture.

Event data is obtained from the surrounding software by means of adapters which are developed considering specifics a practical application. Received event data items are pushed into a queue forming the event data stream. Then the data items are persisted in the event data storage and put into a temporary storage called recent event data. Keeping recent event data separately has the goal of ensuring quick access. Online and offline event data processing are implementations of the speed and batch layers respectively.

Despite of the fact that the lambda architecture pattern is mostly used in the "Big Data world", an approach based on its idea can bring benefits for a so-called "small" data solution like the one described in current paper. The benefit is that it makes possible to transform current version of RTBPM (without significant changes in the implemented event data processing method) so that a new version supports Big Data.

Technology stack

The technologies of RTBPM have been chosen according the criteria: (a) a mature open source solution with good support supplied by its vendor and/or community; (b) ability to scale depending on the actual load; (c) supporting of the Linux operating system. Main technologies of current version of RTBPM are listed in Table 1 (for comparison, the technology stack of the earlier version of system is described in [8],[7]). Most of the technology choices are straight forward and do not require explanations, but the reasons behind some of the not so obvious decisions are briefly commented in the rest of current section.

Process flow visualization (one of the core frontend features) is implemented by means the d3js library. The reason of choosing the d3js library is that it supplies powerful data visualization capabilities based on SVG technology [27]. The server-side components (on the microservices and data processing layers) are mostly

Java-based. The main advantage of choosing Open JDK (instead of Oracle JDK) is the possibility to relay rather on the Java community than on a vendor's implementation. As events are time-stamped data items, InfluxDB (one of the most popular time series databases [28]) has been selected to store historical event data. It should be noted that using InfluxDB is an experimental decision which effectiveness is going to be proved (of disproved) on the further stages of the project. Currently Camunda DMN is applied to specifying business rules for the alerting feature; however, in the future using Camunda is going to be expanded to the conformance checking [3] functionality.

Table 1. Technology stack of RTBPM

Component or RTBPM	**Technology**	**Version**	**Official web site**
Frontend	Angular	6 or higher	https://angular.io
	d3js	5.7.0	https://d3js.org
Microservices	Java / Open JDK	10 or higher	http://openjdk.java.net
	Spring Boot	2.0.4	http://spring.io
Event data stream	Apache ActiveMQ	5.15.5	http://activemq.apache.org
Online event data processing	Java / Open JDK	10 or higher	http://openjdk.java.net
Offline event data processing	Java / Open JDK	10 or higher	http://openjdk.java.net
Recent event data	Redis	4.0.11	https://redis.io
Event data storage	InfluxDB	1.6.2	https://influxdata.com
Process models	MongoDB	4.0.2	https://mongodb.com
Business rules	Camunda DMN	7.9.0	https://camunda.com

PROCESS DISCOVERY METHOD

Choosing of the basic algorithm

The algorithm behind the process discovery method of RTBPM belongs to the process mining field. During last two decades a set of process mining algorithms have been created and examined in practice against real-life data. One of the earliest and simplest technique is the alpha algorithm [2]. It takes event data and produces a Petri net. From practice standpoint the biggest disadvantage of the alpha algorithm is that the generated Petri nets are quite complex (mostly due to a big amount of activities and transitions among them) and consequently are hardly readable for the end users. More advanced in comparison with the alpha algorithm is Fuzzy Miner [13]. Its strong side is the ability to represent a complex process flow model on a certain level of abstraction hiding less important activities and transitions. Practice efficiency of the Fuzzy Miner algorithm proves the fact that it has been adopted by Disco [14] and Celonis [21]. Heuristic Miner [23] is another process mining technique which like the Fuzzy Miner takes into account importance of activities and transitions so that less important elements are not incorporated into a produced process model. Except the version for offline process mining there are Heuristic Miner modifications which aims streaming event data [11],[10]. Some additional info regarding comparison of process mining algorithms can be found in [20].

According to the requirements to the process flow discovery method (see section 3.3) the most suitable for the current version of RTBPM is the Heuristic Miner for stationary event data streams [11].

Implementation of the method

The implemented algorithm is an adaptation of the Heuristic Miner to the lambda architecture. Current version aims stationary event data streams and does not support evolving streams. From the high-level standpoint, the streaming process discovery method of RTBPM consists of the steps:

1. Adapters receive events from the surrounding world, convert them to the XES format [25], and pass to the message queue.
2. Event data items are persisted into the event data storage and passed to the online processor.
3. The online processor maintains the three collections: (a) the most recently received items with weights of their importance; (b) the most recent items for each case; (c) the most recent direct transitions with weights of their importance. The recent event data component is accountable for keeping these collections.
4. The online processor triggers the Heuristic Miner algorithm when it is necessary to update a process model according newly received event data.
5. The frontend part is refreshed automatically (without manual reload the web page) by receiving notification about updates of the process model by means of the Web Sockets [29] technology.

In terms of the lambda architecture steps 1-5 represent flow of the speed layer. In turns, the batch layer (offline event data processing) is executed consuming recent and historical event data. As running of the batch event data processing is resource consuming it is triggered relatively infrequently when the process model requires significant updates.

Further details about implementation of the streaming process discovery method can be found in [9].

An experiment

The implemented streaming process discovery method has been tested on the Road Traffic Fine Management Process [17]. Metrics of the dataset are represented in Table 2.

Table 2. Metrics 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	11
Start date	01 Jan 2000
End date	18 Jun 2013

Control-flow model of the process has been generated by RTBPM using the artificially simulated stream of events from the sample dataset [17].

Two examples of the model snapshots are represented on Fig. 5 and Fig. 6 (the figures have been redrawn manually, because currently the visual models produced by RTBPM are not compact enough to be put into a publication).

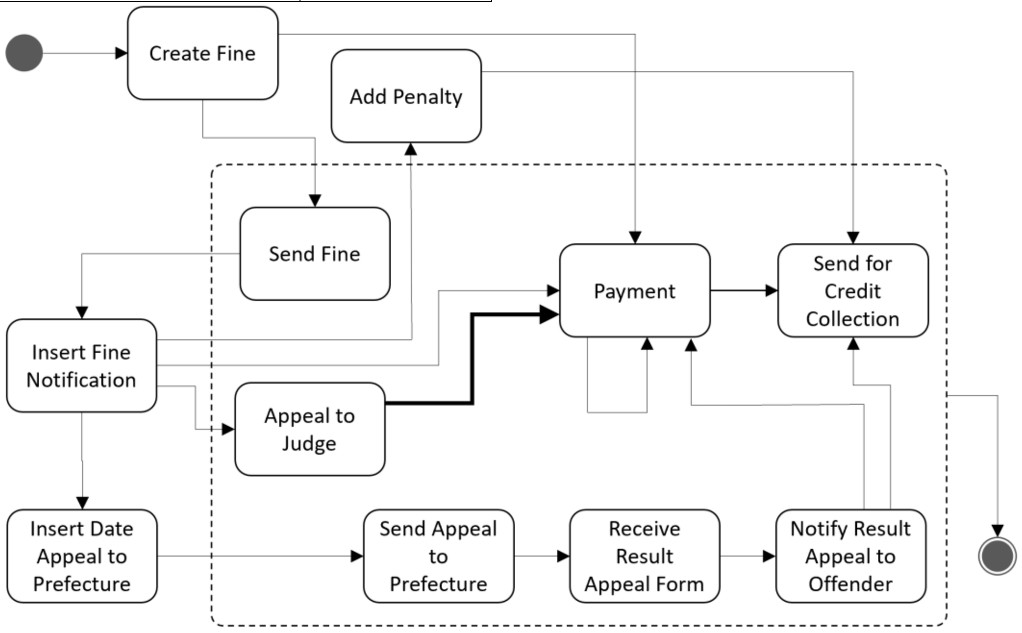
Create Fine
Add Penalty
Send Fine
Payment
Send for Credit Collection
Insert Fine Notification
Appeal to Judge
Insert Date Appeal to Prefecture
Send Appeal to Prefecture
Receive Result Appeal Form
Notify Result Appeal to Offender

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

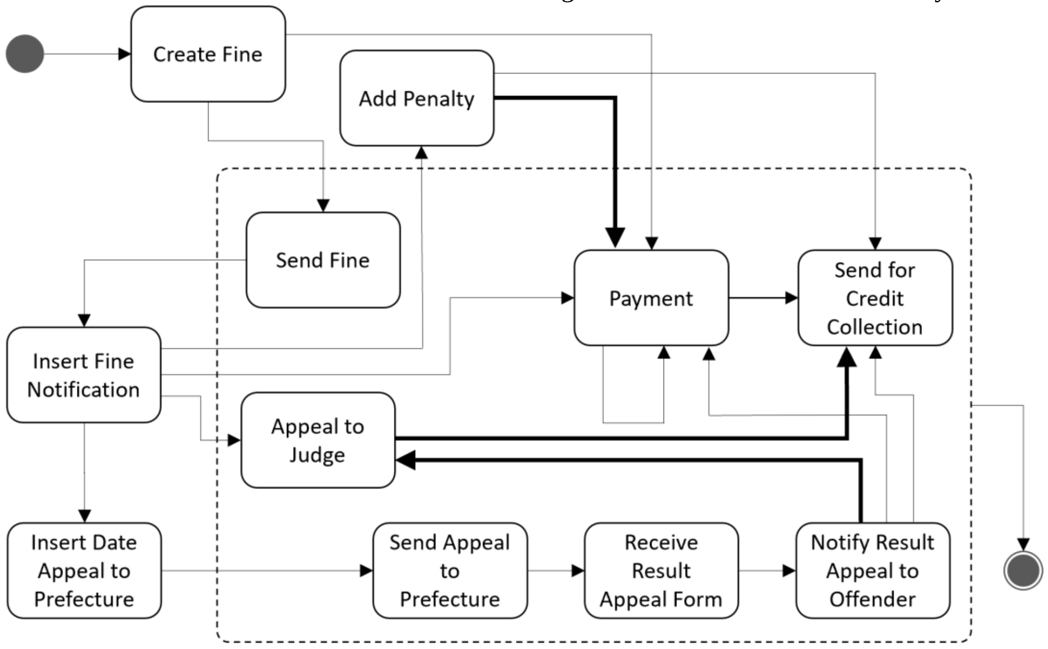
Create Fine
Add Penalty
Send Fine
Payment
Send for Credit Collection
Insert Fine Notification
Appeal to Judge
Insert Date Appeal to Prefecture
Send Appeal to Prefecture
Receive Result Appeal Form
Notify Result Appeal to Offender

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

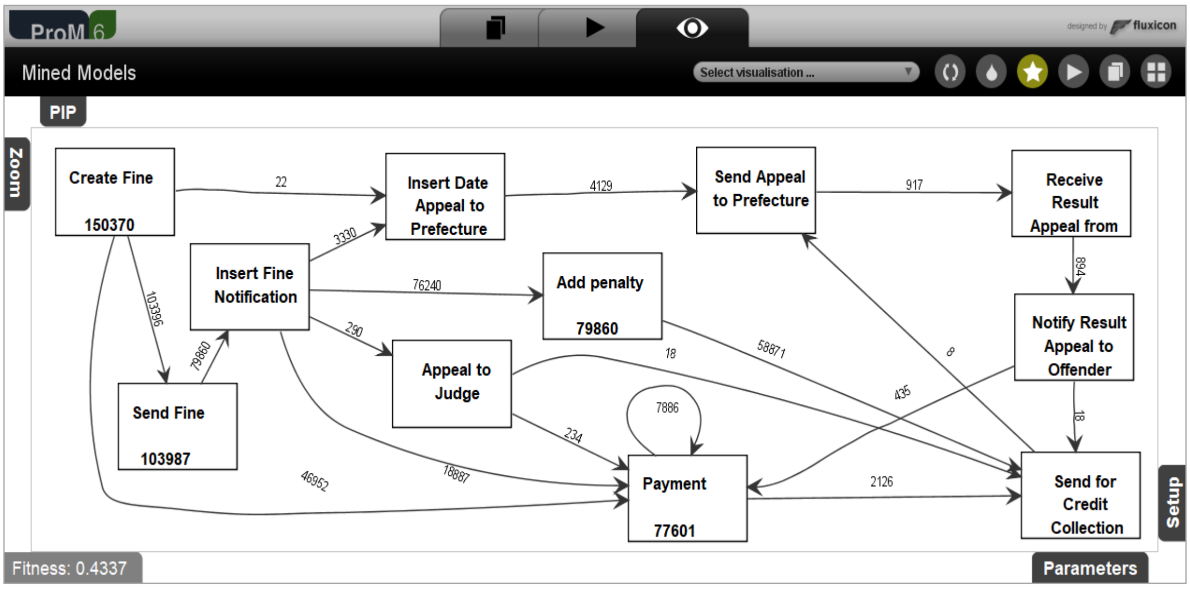

Fig. 7. Control-flow model of the Road Traffic Fine Management Process produced by the Flexible Heuristics Miner plug-in of ProM 6.7

To compare the results produced by the implemented platform the Flexible Heuristic Miner [22] plug-in of ProM 6.7 has been executed against the entire sample dataset. As can be seen from comparison of Fig. 5, Fig. 6, and Fig. 7 the model created by ProM is quite close to the outcomes produced by RTBPM (it is not expected that the three control-flow models are identical because each of them represents state of the real-life process for different timeframes).

SUMMARY AND CONCLUSIONS

Nowadays big amount of organizations suffers from lack of traceability of performed business processes. The problem is especially critical when complex flows, which require interaction of many people and cross a few software systems, are performed in frequently changing business environment where quick response and flexibility are crucial to success.

The designed and implemented by the authors information technology supplies functionality of real-time business process monitoring. The flow chart visualization feature distinguishes RTBPM from classical BI dashboarding solutions which usually do not take into account process origination of the analyzed data. The implemented software is intended to be connected with IT ecosystem of an organization so that it receives event data as well as exposes an integration API which can be consumed by other systems. Since in practice specific requirements to real-time business processes monitoring can variate in a wide range RTBPM is designed as a pluggable platform with possibility to be extended according to the needs of an organization. Another important peculiarity of the represented software is that it is built with free components which allows to avoid additional licensing fees.

Adaptation of streaming version of the Heuritic miner algorithm [11] to a lambda architecture-based [26] system has been used for the first time by the authors. The implemented streaming process discovery method has demonstrated satisfactory results on a test dataset [17]. As shown in section 5.3 the models produced by RTBPM match the model generated by the Flexible Heuristic Miner plug-in of ProM 6.7 [22].

One of the experimental improvements introduced in the RTBPM architecture is using a time series database [28] to store event data. It is expected increasing performance of queries to the event data storage and consequently more efficient work of the batch event data processing. Proving (or disproving) benefits of such a decision is planned for next steps of the project.

Obtaining event data and converting it to the form acceptable for the process mining techniques is one of the most difficult part of connecting RTBPM with an organization’s IT ecosystem. So, there is a risk that such kind of difficulties may become serious impediment for success of an integration project. One of the approaches to mitigate that risk is to implement adapters for some widely-spread software such as SAP, MS Dynamics 365, Atlassian Jira, GitHub etc.

The nearest steps of the project are going to be concentrated on advancing the currently implement streaming process discovery method so that it supports concept drift [15] of the process model. Another task reserved for the future releases is to introduce the conformance checking [3] functionality which makes possible automatic comparison of the real-life and predefined processes. To further the project from the strategic perspective it worth to consider launching a Big Data version of RTBPM and also implementing multitenancy support to make possible SaaS (System as a Service) distribution model.

REFERENCES

1. **Van der Aalst, W.M.P. 2016**. Process mining: data science in action, 2nd ed. Berlin: Springer.
2. **Van der Aalst, W.M.P., Weijters, T., Maruster, L. 2004.** Workflow mining: discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering, 16(9), 1128-1142.
3. **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. Lecture Notes in Business Information Processing, vol. 99. 169-194. Berlin-Heidelberg: Springer.
4. **Bass, L., Clements, P., and Kazman, R. 2012**. Software Architecture in Practice, 3rd ed. Addison-Wesley Professional, 120 p.
5. **Barbacci, M., Klein, M. H., Longstaff, T.H., Weinstock, C.B. 1995.** Quality Attributes (Report No. CMU/SEI-95-TR-021). SEI at Carnegie Mellon University, Pittsburgh, Pennsylvania. Retrieved from https://resources.sei.cmu.edu/asset_files/TechnicalReport/1995_005_001_16427.pdf
6. **Batyuk, A. and Voityshyn, V. 2017**. Business Processes Monitoring by Means of Real-Time Visual Dashboards. In A. Peleschyshyn, O. Markovets (Eds.), The 6th International Academic Conference on Information, Communication, Society. Lviv, Ukraine: Lviv Polytechnic Publishing House.2017. 204-205.
7. **Batyuk, A. and Voityshyn, V. 2018.** Real-Time Process Monitoring Platform: Technical Implementation. In A. Peleschyshyn, O. Markovets (Eds.), The 7th International Academic Conference on Information, Communication, Society 2018 Lviv, Ukraine: Lviv Polytechnic Publishing House. 275-276.
8. **Batyuk, A. and Voityshyn, V. 2018.** Software Architecture Design of the Real-Time Processes Monitoring Platform. In O. Vynokurova, D. Peleshko (Eds.), 2018 IEEE Second International Conference on Data Stream Mining & Processing DSMP’2018. Lviv, Ukraine: Lviv Polytechnic Publishing House. 98-101
9. **Batyuk, A. and Voityshyn V. 2018**. Streaming Process Discovery for Lambda Architecture-based Process Monitoring Platform. In T. Shestakevych (Eds.), 2018 IEEE 13th International Scientific and Technical Conference on Computer Science and Information Technologies. CSIT’2018 Lviv, Ukraine: Vezha and Co. 298-301.
10. **Burattin, A. 2015**. Process Mining for Stream Data Sources. Process Mining Techniques in Business Environments. Lecture Notes in Business Information Processing. Cham: Springer. 177-204.
11. **Burattin, A., Sperduti, A., and van der Aalst, W.M.P. 2012**. Heuristics Miners for Streaming Event Data. CoRR (Computing Research Repository). Retrieved from https://arxiv.org/abs/1212.6383
12. **Fan, W. and and Bifet, A. 2012.** Mining Big Data: Current Status, and Forecast to the Future. SIGKDD Explorations, 14(2), 1-5. DOI: 10.1145/2481244.2481246
13. **Günther C.W. and van der Aalst W.M.P. 2007.** Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics. In: Alonso G., Dadam P., Rosemann M. (Eds.), Business Process Management. BPM 2007. Lecture Notes in Computer Science, 4714 (pp. 328-343). Berlin, Heidelberg: Springer.
14. **Günther, Ch.W. and Rozinat, A. 2012.** Disco: Discover Your Processes. Proceedings of the Demonstration Track of the 10th International Conference on Business Process Management (BPM 2012), Tallinn, Estonia, 940, 40-44.
15. **Jagadeesh Chandra Bose, R. P., and van der Aalst, Wil, M. P., Žliobaitė, I., Pechenizkiy, M. 2011.** Handling Concept Drift in Process Mining. Advanced Information Systems Engineering. CAiSE 2011. Lecture Notes in Computer Science, 6741, 391-405.
16. **Kerremans, M. 2018.** Market Guide for Process Mining. Gartner, Inc. Retrieved from https://www.gartner.com/doc/3870291/market-guide-process-mining
17. **de Leoni, M. and Mannhardt, F. 2015.** Road Traffic Fine Management Process. Retrieved from https://data.4tu.nl/repository/uuid:270fd440-1057-4fb9-89a9-b699b47990f5
18. **Mulesa O., Geche F., Batyuk A., and Buchok V. 2018.** Development of Combined Information Technology for Time Series Prediction. In: N. Shakhovska, V. Stepashko (Eds.), Advances in Intelligent Systems and Computing II. CSIT’2017. Advances in Intelligent Systems and Computing, 689 Cham: Springer. 361-373.
19. **Pathirage, M. 2017.** kappa-architecture.com. Retrieved from http://milinda.pathirage .org/kappa-architecture.com/
20. **Rozinat, A. 2010.** ProM Tips - Which Mining Algorithm Should You Use. Retrieved from https://fluxicon.com/blog/2010/10/prom-tips-mining-algorithm/
21. **Veit, F., Geyer-Klingeberg, J., Madrzak, J., Haug, M., and Thomson, J. 2017.** The Proactive Insights Engine: Process Mining meets Machine Learning and Artificial Intelligence. The 15th International Conference on Business Process Management (BPM 2017). BPM Demo Track and BPM Dissertation Award, Barcelona, Spain, 1920.
22. **Weijters, A. J. M. M. and Ribeiro, J. T. S. 2011.** Flexible Heuristics Miner (FHM). 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Paris, 310-317.

23. **Weijters, A.J.M.M., van der Aalst, W.M.P., and Alves De Medeiros, A.K. 2006.** Process Mining with the Heuristics Miner Algorithm. Eindhoven: BETA Working Paper Series, WP 166.
24. **Business activity monitoring. (2018, Mar 7).** Retrieved from https://en.wikipedia.org/wiki/Business_activity_monitoring
25. **IEEE Standard for eXtensible Event Stream** (XES) for Achieving Interoperability in Event Logs and Event Streams, IEEE Std 1849-2016, 2016.
Lambda Architecture. 2017. Retrieved from http://lambda-architecture.net/
26. **Scalable Vector Graphics** (SVG) 1.1 (Second Edition). (2011, Aug 16). Retrieved from https://www.w3.org/TR/2011/REC-SVG11-20110816/
27. **Time Series Database (TSDB) Explained. (2018).** Retrieved from https://www.influxdata.com/time-series-database/
28. **Web sockets. (2018, Sep).** Retrieved from https://html.spec.whatwg.org/multipage/web-sockets.html