A Practice-Oriented, Control-Flow-Based Anomaly Detection Approach for Internal Process Audits

Gerrit Schumann(✉), Felix Kruse, and Jakob Nonnenmacher

University of Oldenburg, 26129 Oldenburg, NI, Germany
{gerrit.schumann,felix.kruse,jakob.nonnenmacher}@uol.de

Abstract. Internal auditing tries to identify anomalies, weaknesses and manipulations in business processes in order to protect the company from risks. Due to the digitalization of processes, auditors also have to check the associated data volumes. Already existing IT-systems focus on process-related data where the control flow, i.e. the actual sequence of process events, is not considered. This paper examines how the control flow and the process-related data can be analyzed in combination to support auditors in process auditing. To realize this, audit requirements were collected in the literature and evaluated by auditors from industry. On this basis, a concept with five indicators was developed, then transferred into a prototype and evaluated using real-life data as well as two auditors. The results show that the requirements can be technically realized and the developed indicators enable auditors to identify and interpret abnormal process executions.

Keywords: Internal auditing · Process mining · Process context · Unsupervised

1 Introduction

Today, business processes can be controlled and monitored by information systems. Many of these systems can store process events such as transactions or machine signals in a structured form. A collection of such digital events is called *event log*. The basic assumption is that the data in the event log represents the sequence of events as they occurred in reality. In this context, one also speaks of the *control flow* of a process [1].

For internal auditing, the control flow is relevant, since auditors could compare the actual control flow of a process with a target control flow in order to detect anomalies. However, a challenge with such an approach is the availability of a reliable reference model. In practice, these models are often designed at the time the process is introduced, while the actual process flows usually change over time. One reason for such a change could be, e.g., new employees who have a slightly different way of working than their predecessors [2]. To overcome this limitation process mining can be used.

Process mining refers to methods and techniques that can generate a structured process model based on event logs [1]. Using such models, process mining provides the basis for a comparison with an existing reference model or an event log. In contrast to sampling, this would allow auditors to check an entire set of process executions.

E. Kafeza et al. (Eds.): ICSOC 2020, LNCS 12571, pp. 533–543, 2020.
https://doi.org/10.1007/978-3-030-65310-1_39

Despite these potentials, process mining is rarely used in auditing. The reasons for this are suspected, e.g., in the high number of cases classified as anomalous [4]. This typically results from the classical approach of control-flow-based anomaly detection. Most approaches use discovery algorithms to determine a reference model from an event log and then use this model in a conformance check to detect anomalies in the log [5]. However, even a single event, which is not provided in the model can lead to a process execution being interpreted as an anomaly. Such a sensitive interpretation can be a challenge for auditors, as a high number of process executions wrongly interpreted as anomalous can lead to unnecessary investigations. Another aspect is the missing alignment of the technical output with the individual interests of the auditors [4]. The concrete challenge is the atomic form in which the process deviations are provided by the methods [6]. I.e., auditors may see that deviations in the control flow have occurred, but an interpretation of the associated markings is not yet mapped to their specific needs.

To make the output of control flow-based anomaly detection manageable for internal auditing in both quantitative and qualitative terms, we propose an approach that extends the control flow with process-related data and also addresses the auditors' requirements.

1.1 Fundamentals

Event Log and Process Related Data. The event log is essential for process mining and is subject to requirements. One is that each *event* must be assigned to a unique *case*. To be able to consider the ordering, the events must also be assigned to a *timestamp* or sort element. The sequence of events in a case is then called a trace [1]. When a process is running, further data can also be generated, which cannot be assigned to a timestamp or event of the log. However, if this data also has a *case* it can be assigned to the corresponding case in the log. In contrast to the log, which describes the process flow over several lines ①, a single line of process information could be provided in this way ② (Fig. 1).

① Event Log

case	event	timestamp
312P	90	2020-06-01
312P	280	2020-06-02
312P	800	2020-06-03

② Process Related Data

case	amount	class	priority	owner	...	attribute *n*
312P	16	3	high	doe	...	48

Fig. 1. Event log and process related data

Conformance Checking. Conformance checking is a type of process mining and can be used to check whether the real behavior, which is recorded in the log, corresponds to a process model. For the implementation of conformance checks *replays* are used. This term comes from the idea of reproducing or rather "re-playing" the single traces of a log on the process model. As a concrete replay approach, *alignments* have become the standard for conformance checks [1]. Alignments can relate unsuitable events of the log to the process model by a mapping procedure. E.g., let $o = \{90, 265, 160, 280, 467, 492, 413, 496, 730, 769, 810, 820\}$ be a trace of an event log L. If we replay o on the petri net M, this can be visualized in a matrix ①. The bottom row corresponds to the moves of the trace o and the top row to the moves in the petri net M. If a move can be performed both in the trace and in the model, this is called a *synchronous move*. The icon $>>$ indicates a misalignment and appears either when an event recorded in the trace could not be executed in the model (*move-on-log-only*), or when an event that had to be executed by the model did not take place in reality (*move-on-model-only*) [1] (Fig. 2).

①

M	90	265	160	165	280	467	492	>>	496	705	706	730	769	800	810	820
o	90	265	160	>>	280	467	492	413	496	>>	>>	730	769	>>	810	820

② $\text{fitness}(o, M) = 1 - \frac{\text{cost sum of optimal-case alignment } (o)}{\text{cost sum of worst-case alignment } (o)}$

$\text{result} \begin{cases} 0 = \text{no fitness} \\ \cdots \quad \cdots \\ 1 = \text{perfect fitness} \end{cases}$

🟩 synchronous-move
🟪 move-on-model-only
🟨 move-on-log-only
>> misalignment

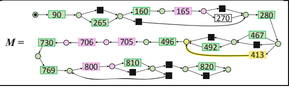
$M =$


Fig. 2. Conformance checking using alignments

During a replay there can be many different alignments between a log and a process model. For the calculation of the replay fitness only the optimal and worst-case alignment is needed. Here, it is necessary to assign costs to the moves. For moves where the log and the model match (synchronous moves), these costs must always be set to 0. If log and model differ and no process knowledge is available, a fixed cost value of 1 can be assigned to all misalignments. To calculate the replay fitness between o and M, the costs of all moves are then summed up for each trace and filled into the formula ② [1].

1.2 Related Work

While most conformance checking methods have focused on the control flow, some studies also considered additional process data. Such data can be used to check whether the control flow also meets the correct process conditions (e.g., correct control flow activity performed by an authorized person). To consider such aspects in the conformance check, in [7] an alignment approach is extended by a heuristic-based procedure. In [8] an approach based on integer linear programming is presented, which, in contrast to [7], is also capable of processing numerical values. In [9] a constraint-based approach for conformance checking of declarative processes is developed. In [10] both the data and the time perspective are linked to the control flow. In [11] a cost function is used to prioritize deviations between a log and a model, considering all process perspectives.

Further studies focused on reducing the high number of control-flow-based alerts or false-positives. E.g., in [12], not every unexpected event is immediately classified as abnormal, but assessed based on the likelihood of its occurrence. In [13] an association rule mining approach is proposed that can be used to determine what behavior might have caused an abnormal process. This can help to differentiate between harmful anomalies and false-positives. Besides the purely technical research, process mining is also investigated in the auditing context [3, 14–17]. These studies stand out from others because they already address concrete tasks and scenarios in the auditing area.

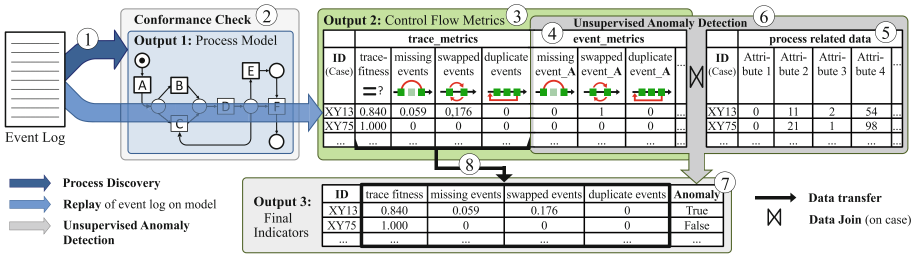
2 Requirements

To determine the requirements for our approach, we conducted a literature research that aimed to identify the criteria according to which an auditor selects anomalous process executions based on their control flow behavior. As a result, only the study by [4] was identified as relevant. This is due to the fact that the known control-flow-related deviation patterns originate from the field of process mining or business process management and until recently these patterns were not checked for consistency with the deviation patterns used by auditors. It was only the study by [4] that provided representative answers by conducting interviews with auditors. According to [4], especially ***swapped***, ***missing*** and ***duplicate*** events in the control flow are criteria that are relevant for auditors. In order to assess these requirements, the following aspects were discussed with two auditors. A1: Compliance of the collected criteria with those of the own audit department, A2: Suitable way of presenting the criteria in concrete figures/indicators.

After aspect A1 was discussed, it can be stated that the requirements were confirmed by the auditors. They added, that it would be helpful if they could also be informed about the (ab-)normality of the associated process conditions when evaluating the control flow deviations. This could help to interpret whether the deviations may be typical or atypical under certain process circumstances. Aspect A2 showed that, a combination of deviation criteria as numerical indicators would be helpful for the auditors. The concrete proposal was a labeling that consists of four indicators. Three of these would be represented by the fields *missing*, *swapped* and *duplicate* events which provide the rates these deviations types have for each process execution. The fourth indicator should be an overall value, which summarizes the conformance of each process execution.

3 Conceptual Design

Our concept consists of two phases. One for the integration of the control flow and a second one for the process-related data. In the first phase, the upstream task is the process discovery ①, which creates a process model based on the event log and a mining procedure. This model represents the basis for the second task: the conformance check ②, which aims to measure the deviations between the log and the process model by *replaying* the log on the model. The conformance check provides the indicators that show the rates of missing, swapped and duplicate events as well as the overall conformity for each process execution (trace). These indicators will be calculated both for each event of the trace (*event_metrics*) and for the entire trace (*trace_metrics*) ③ (Fig. 3).


Event Log
1
Conformance Check
2
Output 1: Process Model
A
B
C
D
E
F
Output 2: Control Flow Metrics
3
Unsupervised Anomaly Detection
6
4
5
=?
⋈
8
7
Output 3: Final Indicators
Process Discovery
Replay of event log on model
Unsupervised Anomaly Detection
Data transfer
Data Join (on case)

	trace_metrics				event_metrics			
ID (Case)	trace-fitness	missing events	swapped events	duplicate events	missing event_A	swapped event_A	duplicate event_A	...
XY13	0.840	0.059	0,176	0	0	1	0	...
XY75	1.000	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...

	process related data				
ID (Case)	Attri-bute 1	Attri-bute 2	Attri-bute 3	Attri-bute 4	...
XY13	0	11	2	54	...
XY75	0	21	1	98	...
...	...	...	...	...	...

ID	trace fitness	missing events	swapped events	duplicate events	Anomaly
XY13	0.840	0.059	0.176	0	True
XY75	1.000	0	0	0	False
...	...	...	...	...	...

Fig. 3. Concept

In this concept, the process discovery only considers the control flow. The resulting process model thus does not contain *guards*. Guards are rules that explain when process executions have to take a certain path at a decision point in the process model [7]. This means, a process model, which is extended by guards (a data-petri-net [18]), describes not only which process paths are possible, but also under which conditions they may be executed. In our concept, the event log could also be used in combination with the process-related data to create a data-petri-net. However, regardless of the process model type, we propose an additional data combination. This is due to the fact that during a conformance check even guards can only control the behavior that is modeled in the process model. However, a process model can never contain the entire behavior of a real business process without becoming unreadable for a person. Instead, process models are typically created in a more general form. If a conformance check is then performed using such a generalized model, the number of deviating process executions can quickly increase when the log is noisy. In order to identify the legitimate – but not modeled – cases within this set of potential anomalies, the control flow deviations can also be subsequently combined with process context. In our approach, this combination is realized by further processing the previously calculated *event_metrics* ④ together with the process related data ⑤ using an unsupervised anomaly detection ⑥. I.e., the data processed by the anomaly detection contains a row for each process execution with the fields: *case* | *missing_event_A* | *swapped_event_A* | *duplicate_event_A* | *missing_event_B* |...| *missing_event_X* |...| *process_related_attr_1* | *process_related_attr_2* |...| *process_related_attr_n*. In this way, it is possible to put the control flow behavior of a process execution into a concrete process context, even if this behavior is not provided in the generalized model (e.g., to determine, whether the missing event *A* is typical or untypical, considering the respective process context). In our concept, this results in an additional indicator ⑦ that provides a boolean value, where "*true*" stands for an abnormal and "*false*" for a normal process execution. The approach is unsupervised, as no labels should be used. The final output also includes the aggregated control flow metrics ⑧.

4 Case Study

Setting, Data and Preprocessing. To evaluate our concept, we transferred it into a prototype and conducted a case study in the internal auditing department of an internationally operating automobile manufacturer. In addition to the expertise of two auditors the case study benefited from real-life data from 2058 manufacturing-related permit processes. The event log contains status numbers as events, which were set in the information systems during a permit process. The process-related data provides single-line information on 54 fields about each permit process without referring to events or times in the control flow. The categorical fields of the process-related data set (8% of the data) were converted into numeric values using a frequency-based [19] encoding. A one-hot encoding [20] was omitted because in this data, this would have generated tens of thousands of columns and thus greatly increased the object space dimensionality. Afterwards, a min-max normalization (0, 1) has been performed for all attributes.

Process Discovery. To perform the process discovery and conformance checking, we used *ProM*, an open source framework for process mining algorithms [21]. For the process discovery, the plugin *Inductive Visual Miner* [22] and for the process representation the petri net was used. The Inductive Visual Miner can be adjusted using the *path slider* and the *activities slider*. The latter determines the proportion of events to be considered in the process model. The path slider can be used to control the scope of the noise filtering and ranges, just like the activity slider, from 0 to 1 (0 = maximum noise filtering). In this context "noise" refers to the variation of paths between the events [22].

Since initially no events should be excluded from the log, the activities slider was set to 1 during the entire process discovery. In contrast, the path slider (*ps*) was examined at 21 settings (0.0, 0.05, 0.10,..., 1.0). To simplify the creation of a generalized model, these 21 settings were used for the evaluation of 10 chaotic event filtering [23] and 20 trace filtering [24] iterations. E.g., during the chaotic event filtering, one chaotic event after another was successively filtered out of the log and after each filtering, the 21 *ps*-positions were iterated. Since one process model results from each path slider setting, a total of 630 process models were created in this way (21 *ps* * 10 + 21 *ps* * 20). By replaying the log on each of these process models, the *fitness* and *precision* as well as the *f-score* were calculated. After running all iterations, the model with the maximum f-score was selected for the prototype (f-score: 97.6 | fitness: 96.9 | precision 98.3).

Conformance Check. We used the calculation of alignments in our conformance check. Since the previously created process model is considered as a reference process and the single process executions (traces) should be assessed with regard to their conformity to this reference process, the perspective of *trace fitness* takes effect. To address this perspective through alignments we applied the plugin *Conformance Checking of DPN (XLog)* [11] which used the process model and the log as input. The output is a list of all traces, in which the control flow violations are marked by different coloring. Figure 4 shows two traces of the log: one with 100% trace fitness ① and one with 81.8% ②.

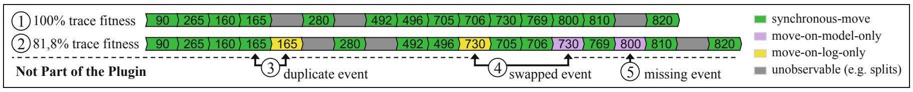


Fig. 4. Conformance checking output

Given the collected requirements to detect missing, swapped, and duplicate events, this output needs additional interpretation. The purple and yellow markings indicate whether the event could be set at the respective position exclusively by the model (move-on-model-only) or by the log (move-on-log-only). For the alignment calculation, these move classifications are performed exclusively for the considered position in the trace and model. However, the fact that, e.g., a move-on-model-only needs to be interpreted as a completely missing event, would only be true if this event was not set at any other position in the trace. Such an actually missing event is shown in ⑤. If, on the other hand, the event would also be set at an earlier or later position in the trace, this should be interpreted as a swapped event ④. The move-on-log-only shows that the event was not intended by the model. Besides a swapped event, this could also be due to a duplicate event ③. Such a global consideration of the entire trace is irrelevant for the fitness calculation and is thus not provided by the plugin. To calculate the *missing*, *swapped* and *duplicate events*, we therefore implemented a script which used the csv-export-file of the plugin as input. In this file, the move type is given by numbers: **0** (*synchronous-move*), **1** (*move-on-log-only*), **2** (*move-on-model-only*). Our script stores these values together with the occurring event names for each trace in three lists (type_0, type_1, type_2). Then it iterates over the lists and derives the deviation categories for each event based on the logic shown in ⑥. In this way, each trace event receives its own deviation categories (*event_metrics*). E.g., for event 730, the columns *730_duplicate, 730_swapped*, and *730_missing* would be created in which the respective frequencies are stored ⑦. In a final aggregation, the values of the *event_metrics* are set in relation to the number of all events occurring in the trace (for trace ② of Fig. 4: 1/14 events = 0.0714 ⑧) (Fig. 5).

Calculation Logic for the *event_metrics* ⑥	
Action	Condition (**MT** = move type; **e** = the considered event within the trace)
duplicate+1	A synchronous-move (**MT**: 0) and a move-on-log-only (**MT**: 1) was found for **e**
swapped+1	A move-on-log-only (**MT**: 1) and a move-on-model-only (**MT**: 2) was found for **e**
missing+1	Only a move-on-model-only (**MT**: 2) was found for **e**

Output of the Calculation									
	⑧ trace metrics					⑦ event metrics			
Case	trace fitness	duplicate events	swapped events	missing events	...	730_ duplicate	730_ swapped	730_ missing	...
XY23	0.8181	0.0714	0.0714	0.0714	...	0	1	0	...

Fig. 5. Calculation of the control flow indicators

Unsupervised Anomaly Detection. We used two views of the process-related data. In the first view no attributes were filtered out (*data_view*). The second view included, besides the *event_metrics*, only attributes in which our auditors found anomalies (*aud_view*).

For the unsupervised anomaly detection, performance between an iForest [25] and a local outlier factor (lof) [26] was compared. The lof classifies a data point as (ab)normal based on the local density deviation of the data point in relation to its neighbors [26]. The iForest uses a combination of several isolation trees, which isolate outliers from the rest of the data by a recursive, random division of the attribute values [25]. These two methods were implemented using the python library *scikit-learn* which explains the available parameters in [27]. For these parameters, different value intervals were defined and combined with one another. This resulted in 400 iForest and 480 lof parameter settings. Since no process knowledge was assumed in our concept, an internal evaluation of the model quality was necessary. In this way, the influence of the different parameter settings can be measured without using ground truth labels. To do so, we used a classifier performance approach. This assumes that if an anomaly detection method efficiently captures a data set, the abnormal entities identified by this method must be well separated from the normal entities. This separability can in turn be measured by a supervised classification. The goal is to capture the prediction quality of a classifier, which used the previously predicted anomaly classes of the unsupervised anomaly detection as target labels during the training [28]. If the classifier is able to predict the outliers in a test set well, this indicates a good separability. To measure such a classifier performance, we implemented a loop that performed an unsupervised anomaly detection for each of the 400 (iForest) and 480 (lof) parameter settings as well as a subsequent supervised classification on the anomaly detection results. Since the unsupervised anomaly detection did not require training, it was applied to the entire data set. For the subsequent supervised classification, a random forest was applied which used 60% of the data for training and

40% as a test set to measure the prediction quality. To guarantee the same conditions for each iteration, all random variables (in train/test-split, random forest, iForest) were set to 0. As a result of each loop iteration, the classification quality was measured using the f-score and stored in a list. After the loop terminated, the maximum f-score was determined from the list and the corresponding model and parameter setting was used for the final anomaly detection. For both the *data_view* (f-score: 0.936) and *aud_view* (f-score: 0.998) the maximum f-score belongs to the iForest.

4.1 Evaluation

As a first step of this evaluation, the auditors of the partner company converted their audit findings into binary anomaly flags ①. Based on the comparison of these flags with those of the prototype ②, two confusion matrices were calculated. These matrices show differences regarding the hit rate of the actually (ab)normal cases. While in the *data_view* ③ 422 anomaly classifications (20.5%) deviated from the audit report, only 249 (12.1%) deviated in the *aud_view* ④. The results also show that the prediction of actually abnormal cases is more accurate if the set of attributes is limited by auditors (Fig. 6).

	Prototype Indicators					②		① **Audit Report**	
	Case	trace fitness	missing events	swapped events	duplicate events	anomaly		anomaly	Case
⑥	XY31	1.0	0.0	0.0	0.0	true	⟷	true	XY31
	PW24	0.850	0.0	0.166	0.083	true	⟷	true	PW24
⑤	RT67	0.944	0.0	0.060	0.0	false	⟷	false	RT67
	...	...	...	...	...	...	⟷	...	...

③ **Confusion Matrix – Data View**			
Deviations: 422		Audit Report	
		abnormal	normal
Proto-type	abnormal	339	287
	normal	135	1297

④ **Confusion Matrix – Aud View**			
Deviations: 249		Audit Report	
		abnormal	normal
Proto-type	abnormal	446	221
	normal	28	1363

Fig. 6. Comparison of the anomaly flags of the prototype with those of the audit report

In a second step, the auditors assessed the indicators in functional terms. As a result, it can be stated that they consider the indicators as a useful audit instrument. This perception is based on the ability to sort and group the process executions by the indicators values. Since the fifth indicator considers every event deviation in its respective process context, auditors can also see when a deviation is relativized by its context. In such a case a control flow deviation would be displayed, but the anomaly flag would be *false* ⑤. The added value of such a relativization becomes clear if auditors would use only the four control flow indicators to select potential audit cases. The auditors would then classify a process execution as relevant for an audit if it deviates from the model (e.g. fitness < 1). If we had used such an exclusively control-flow-based approach for our comparison with the audit report, there would be 7.3% more false-positives. It is also possible to select only those cases that have been classified as abnormal solely because of the process-related data conditions. I.e., no control flow deviation is displayed but the anomaly flag is set to *true* ⑥. The combined view of the four control flow indicators together with the anomaly flag thus allows auditors to better interpret an anomaly.

In a final step, the control flows of the false-positives were examined in detail, as they had a much larger proportion compared to the false-negatives. It was found that the deviations in 76,1% of all false-positives were due to events which, according to the auditors, are known to be chaotic in nature. This includes, e.g., the responses from email distribution lists. In some cases, several departments must provide a statement on a requested permit process. However, the times at which these responses are made can be very arbitrary – which is legitimate. It is only important that all responses are received before proceeding to the next approval step. This implies that some *switched_events* are legitimate in this case study. If departments do not respond, the requests will be sent again. This is also legitimate and leads to *duplicate_events*. The reason why this behavior was not considered by a loop in the process model is that these cases did not occur often enough to be considered by the process discovery method (when using maximum f-score). Given this legitimate and at the same time negligible process variance, the concerned 76,1% were confirmed as actual false-positives. In the remaining 23,9% alleged false-positives, specific control flow deviations were found which were actually relevant for further audits.

5 Conclusion

In this paper we investigated how a control-flow-based anomaly detection can be used to support auditors in selecting audit-relevant process executions. An important feature of this approach is its intended practical suitability. In addition to the conceptual alignment with the audit-requirements, this claim should also be achieved through a practice-oriented data requirement. Since unlabelled data are common in practice, no labelled data were used during the modeling. The entire procedure was thus realized without process knowledge. The evaluation of the developed indicators confirmed that these can provide auditors with useful assistance in the selection of relevant cases. In addition to the evaluation results, this is also due to the comparison with classical audit methods, which are still characterized by sampling and focusing on process-related data. The practical application of our approach is therefore not only the consideration of a full data population, but also the use of both the control flow and the process related data. In this way it was also shown that control flow deviations can be relativized by their individual context. This in turn can help to reduce the amount of irrelevant cases.

For theoretical application, this paper provides new insights in terms of integrating data-driven methods in internal auditing. Three aspects are crucial in this context: 1) Our approach addresses the current reasons that hinders the establishment of control-flow-based anomaly detection in internal auditing. 2) In addition to a concept, our paper presents an exemplary implementation. 3) The suitability of our approach is evaluated quantitatively using real-life data and qualitatively by two auditors.

A limitation of our approach is that the consideration of process-related data could only reduce the false-positives to a certain extent, but not entirely. I.e., auditors could still be held up examining some irrelevant cases. In future work, we want to overcome this limitation and also test the approach in further real-world auditing scenarios.

References

1. Van der Aalst, W.M.P.: Process Mining: Data Science in Action, 2nd edn. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
2. Nolle, T., Seeliger, A., Mühlhäuser, M.: Unsupervised anomaly detection in noisy business process event logs using denoising autoencoders. In: Calders, T., Ceci, M., Malerba, D. (eds.) DS 2016. LNCS (LNAI), vol. 9956, pp. 442–456. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46307-0_28

3. Jans, M., Alles, M., Vasarhelyi, M.: A field study on the use of process mining of event logs as an analytical procedure in auditing. Account. Rev. **89**(5), 1751–1773 (2014)
4. Hosseinpour, M., Jans, M.: Process deviation categories in an auditing context. Available at SSRN 3280339 (2019)
5. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: Analyzing business process anomalies using autoencoders. Mach. Learn. **107**(11), 1875–1893 (2018). https://doi.org/10.1007/s10994-018-5702-8
6. Hosseinpour, M., Jans, M.: Categorizing identified deviations for auditing. In: SIMPDA 2016, Graz, Austria. pp. 125–129 (2016)
7. de Leoni, M., van der Aalst, W.M.P., van Dongen, B.F.: Data- and resource-aware conformance checking of business processes. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds.) BIS 2012. LNBIP, vol. 117, pp. 48–59. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30359-3_5
8. de Leoni, M., van der Aalst, W.M.P.: Aligning event logs and process models for multi-perspective conformance checking: an approach based on integer linear programming. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 113–129. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40176-3_10
9. Borrego, D., Barba, I.: Conformance checking and diagnosis for declarative business process models in data-aware scenarios. Expert Syst. Appl. **41**(11), 5340–5352 (2014)
10. Burattin, A., Maggi, F.M., Sperduti, A.: Conformance checking based on multi-perspective declarative process models. Expert Syst. Appl. **65**, 194–211 (2016)
11. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Balanced multi-perspective checking of process conformance. Computing **98**(4), 407–437 (2015). https://doi.org/10.1007/s00607-015-0441-1
12. Böhmer, K., Rinderle-Ma, S.: Multi-perspective anomaly detection in business process execution events. In: Debruyne, C., Panetto, H., Meersman, R., Dillon, T., Kühn, e, O'Sullivan, D., Ardagna, C.A. (eds.) OTM 2016. LNCS, vol. 10033, pp. 80–98. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48472-3_5
13. Böhmer, K., Rinderle-Ma, S.: Association rules for anomaly detection and root cause analysis in process executions. In: Krogstie, J., Reijers, Hajo A. (eds.) CAiSE 2018. LNCS, vol. 10816, pp. 3–18. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91563-0_1
14. Alizadeh, M., Lu, X., Fahland, D., Zannone, N., van der Aalst, W.M.P.: Linking data and process perspectives for conformance analysis. Comput. Secur. **73**, 172–193 (2018)
15. Van der Aalst, W.M.P., van Hee, K. M., van Werf, J.M., Verdonk, M.: Auditing 2.0: using process mining to support tomorrow's auditor. Computer **43**(3), 90–93 (2010)
16. Jans, M., Alles, M., Vasarhelyi, M.: Process mining of event logs in internal auditing: a case study. In: 2nd International Symposium on Accounting Information Systems, Rome (2012)
17. Barboza, T.M., Santoro, F.M., Revoredo, K.C., Costa, R.M.M.: A case study of process mining in auditing. In: Proceedings of the XV Brazilian Symposium on Information Systems, pp. 1–8 (2019)
18. De Leoni, M., van der Aalst, W.M.P.: Data-aware process mining: discovering decisions in processes using alignments. In: Proceedings of the 28th Annual ACM Symposium on applied Computing, pp. 1454–1461. ACM Press (2013)
19. Das, S., Cakmak, U.M.: Hands-On Automated Machine Learning: A Beginner's Guide to Building Automated Machine Learning Systems Using AutoML and Python. Packt Publishing, Birmingham (2018)
20. Chollet, F.: Deep Learning with Python. Manning Publications, Shelter Island (2017)
21. ProM. http://www.processmining.org/prom/start. Accessed 19 Apr 2020
22. Leemans, S.: Inductive visual Miner & Directly Follows visual Miner – manual (2019)
23. Tax, N., Sidorova, N., van der Aalst, W.M.P.: Discovering more precise process models from event logs by filtering out chaotic activities. J. Intell. Inf. Syst. **52**(1), 107–139 (2019)

24. Conforti, R., La Rosa, M., ter Hofstede, A.H.M.: Filtering out infrequent behavior from business process event logs. IEEE Trans. Knowl. Data Eng. **29**(2), 300–314 (2017)
25. Liu, F.T., Ting, K.M., Zhou, Z.-H.: Isolation forest. In: Proceedings of the Eighth IEEE International Conference on Data Mining, pp. 413–422. IEEE Computer Society (2008)
26. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. ACM Sigmod Rec. **29**(2), 93–104 (2000)
27. Sklearn. https://scikit-learn.org/stable/modules/outlier_detection.html. Accessed 24 Apr 2020
28. Nguyen, T.T., Nguyen, A.T., Nguyen, T.A.H., Vu, L.T., Nguyen, Q.U., Hai, L.D.: Unsupervised anomaly detection in online game. In: Proceedings of the Sixth International Symposium on Information and Communication Technology, pp. 4–10, ACM (2015)