Detecting Complex Anomalous Behaviors in Business Processes: A Multi-perspective Conformance Checking Approach

Azadeh Sadat Mozafari Mehr(✉), Renata M. de Carvalho, and Boudewijn van Dongen

Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands
{a.s.mozafari.mehr,r.carvalho,b.f.v.dongen}@tue.nl

Abstract. In recent years, organizations are putting an increasing emphasis on anomaly detection. Anomalies in business processes can be an indicator of system faults, inefficiencies, or even fraudulent activities. In this paper we introduce an approach for anomaly detection. Our approach considers different perspectives of a business process such as control flow, data and privacy aspects simultaneously.Therefore, it is able to detect complex anomalies in business processes like spurious data processing and misusage of authorizations. The approach has been implemented in the open source ProM framework and its applicability was evaluated through a real-life dataset from a financial organization. The experiment implies that in addition to detecting anomalies of each aspect, our approach can detect more complex anomalies which relate to multiple perspectives of a business process.

Keywords: Outlier behavior detection · Anomalous behavior · Data privacy · Conformance checking · Multi-perspective analysis

1 Introduction

Today, anomaly detection is essential for businesses. This concept refers to the problem of finding patterns in data that do not conform to regular behavior. Outliers and anomalies are two terms commonly used in regards to anomaly detection. The importance of outlier or anomaly detection lies in the fact that anomalies in data can be translated into valuable, and often critical and actionable information in a variety of applications such as fraud detection, intrusion detection for cyber-security, and fault detection in systems [4]. In the business process management domain, anomaly detection can be applied for detecting anomalous behaviors during business processes executions. Often, organizations look for anomalies in their business processes, as these can be indicators for inefficiencies, insufficiently trained employees, or even fraudulent activities. Mostly, companies rely on process-aware information systems to manage their daily processes. The event logs of these information systems are a great source of information capturing executed behavior of different elements involved in the business processes such as

M. Montali et al. (Eds.): ICPM 2022 Workshops, LNBIP 468, pp. 44–56, 2023.
https://doi.org/10.1007/978-3-031-27815-0_4

employees and systems. They can be used to extract valuable information about the executions of a process (process instances) as they reflect executed behaviors. In the context of business processes, an anomaly is defined as a deviation from a defined behavior, i.e., the business process model [11].

Nowadays, business processes have a high level of complexity. On top of a daily process, many standards and regulations are implemented as business rules which should be considered in anomaly analysis. For compliance checking, business analysts should investigate the processes from multiple perspectives. This is a very challenging task since different aspects of processes should be considered in both isolating and combining views in order to detect hidden deviations and anomalous behaviors. For instance, generally employees are authorized to access sensitive data only in the context of working and for a defined purpose. Privacy violations may happen when employees misuse this authority for secondary purposes like personal or financial benefits. In this regards, one of the articles in the GDPR regulation is about purpose limitation emphasizing "Who can access data for which purpose?". Such data privacy rule is closely related to three different perspectives: i) the control flow, or the tasks being executed; ii) the data, or the flow and processing of information; and iii) the privacy, or the legitimate role allocation. This example clearly shows that the approaches which focus only on control flow or data flow aspects are not sufficient to detect deviations and anomalies in complex problems. The potential of multi-perspective process mining has been emphasized by several contributions [2,6,8]. Although these techniques consider data objects and/or the resources, in all of them control flow is a priority since they assume data objects or resources as attributes of activity instances in the process execution.

Previously, we presented a balanced multi-perspective approach for conformance checking and anomaly detection which considered control-flow, data and privacy perspectives all together and simultaneously without giving priority to one perspective [9]. In this paper, we extend our previous approach by considering the type of data operations (mandatory or optional) and their execution constraints in the calculation of alignments. To the best of our knowledge, no other approach takes data layer restrictions of data operation type and frequency into account. Furthermore, in our new approach, we made the concept of context (purpose) of data processing more clear. As another improvement, to avoid reporting false positive deviations in the control flow perspective, we consider partial order of activity executions. Similarly, Lu *et al.* [7] used partial order in event data to improve the quality of conformance checking results. However their approach checks only control flow alignment in contrast to our approach which is a multi-perspective conformance checking method.

The remainder of this paper is structured as follows. Section 2 introduces our multi-perspective conformance checking approach to detect complex anomalous behaviors in business processes. Section 3 presents the applicability of our approach through a real-life case study, discussing the experimental design and results. At last, the conclusion of this paper is presented in Sect. 4.

2 Methodology

Current conformance checking methods use alignments (in detail explained in [3]) to relate the recorded execution of a process with its model. Commonly, these techniques have a fundamental property, so-called synchronous product model. A synchronous product model links observed behavior and modeled behavior in a Petri net format. By using an A* based search strategy [1, 12], the conformance checking techniques can compute alignments for individual cases in an event log.

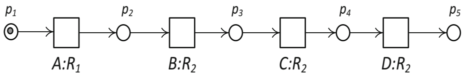
While traditional conformance checking approaches only consider control flow aspect of a process, we consider data and privacy aspects together with control flow perspective all at once. In the rest of this section, we explain the structure of the synchronous product model in our new approach for multi-perspective conformance checking and show the types of anomalies that our method is able to detect by employing A* algorithm on the designed synchronous product model.

2.1 Construction of Synchronous Product Model

To clarify the steps of constructing the synchronous product in our approach, let us consider the inputs shown in Fig. 1. Figure 1(a) shows a workflow-net as the process model. This process model starts with activity *A* by role R1 and continues with activities *B*, *C*, and *D* by role *R2*. According to the data model depicted in Fig. 1(b), for the completion of activity *A*, mandatory data operation *Read(x)* should be executed and the actor is allowed to repeat this data operation. *Update(y)* is another data operation in the context of activity *A* that is optional and the actor is allowed to execute this operation only once while performing *A*. Each of activities *B*, *C*, and *D* are expected to execute one mandatory data operation in order to fulfilment. Figure 1(c) shows the organisational model in our example. There are two roles in the organisational model. Actor (resource) *u1* has the role *R1* and the actor *u2* has the role *R2*.

Figure 1(d) shows one trace of the process log. This trace contains eight process events that correspond to a single case. The start and complete events with the same activity name and id indicate the occurrence of an instance of a specific activity. For example, e_3 and e_4 both with id equal to 2 indicate the execution of one instance of activity B. The events are sorted by their occurrence time.

Figure 1(e) presents a data trace with three data operations *op1*, *op2*, and *op3*, which were executed on the data fields *x*, *z*, and *m* during the execution of case 100.



(a). Process Model

Activity	Data Operations	Mandatory/ Optional	Repetition
A	d1: Read(x) d2: Update(y)	M O	Allowed Not Allowed
B	d3: Update(z)	M	Not Allowed
C	d4: Update(K)	M	Not Allowed
D	d5: Read(x, y)	M	Not Allowed

(b). Data Model

Role	Actor (Resource)
R1	u1
R2	u2

(c). Organizational Model

CaseID	EventID	ID	Activity	Actor	Type	Time
100	e1	1	A	u1	Start	2021-02-01 12:38:44
100	e2	1	A	u1	Complete	2021-02-01 12:58:44
100	e3	2	B	u1	Start	2021-02-01 12:58:44
100	e4	2	B	u1	Complete	2021-02-01 13:39:38
100	e5	3	C	u1	Start	2021-02-01 13:42:38
100	e6	3	C	u1	Complete	2021-02-01 13:48:38
100	e7	4	F	u1	Start	2021-02-01 14:20:38
100	e8	4	F	u1	Complete	2021-02-01 14:36:38

(d). Process Trace (a fragment of the process log)

CaseID	EventID	Operation	Actor	Type	Time
100	op1	Read(x)	u1	Complete	2021-02-01 12:40:44
100	op2	Update(z)	u1	Complete	2021-02-01 13:20:37
100	op3	Update(m)	u1	Complete	2021-02-01 13:32:38

(e). Data Trace (a fragment of the data log)

Fig. 1. The inputs of the proposed approach in the running example

Figure 1(d) together with Fig. 1(e) shape the observed behavior for case 100. A close inspection of the event logs already shows that there are some conformance issues. First, from the control flow perspective, activity *D* appears to be missing while activity *F* is an unexpected activity according to the process model. Second, from data perspective, two mandatory data operations *d4* and *d5* are missing and *op3* implies the execution of a spurious data operation by user *u1*. Third, from privacy (resource) perspective, activities *B* and *C* are expected to be performed by a user playing role *R2*, but it appears that these activities and data operations were performed by user *u1* who plays the role *R1*. From combined perspectives, although activity *B* was performed in correct order and expected by the process model and its executed data operation (*op2*) conforms with the data model, there is a deviation in the privacy aspect. Data operation *op2* is only supposed to be executed within the context of activity *B* by an actor playing the role *R2* however this data operation was accessed by a user who plays the role *R1*.

A traditional conformance checking technique, which focuses only on the control flow, would ignore the resource and data parts of the modeled behavior. To address this issue, now we present our approach which considers control flow, data and privacy aspects of a business process simultaneously for anomaly detection analysis and can automatically distinguish all kind of anomalies which were described earlier.

As a pre-processing step, to combine process, data and privacy (resource) aspects into a single prescribed behavior, we first shape the operation net of each activity in the process model considering corresponding data operations in the data model. For instance, the operation net of activity *A* is depicted in Fig. 2 surrounded by a red line. It represents how we model mandatory and optional data operations and their execution constraint in a Petri net format. In the operation net of an activity *X*, there are two corresponding transitions labelled with "*Xs*" (X+Start) and "*Xc*" (X+complete) (i.e. transition *As* and *Ac* in Fig. 2). For each expected data operation of the activity, one transition labeled with the name of data operation and two places are created: one is the input place and the other is the output place of the expected data operation. The input place of the expected data operation is an output place for the activity transition with the start type, while the output place of the expected data operation is an input place for the activity transition with the complete type. An invisible transition is created and connected to input and output places of each optional data operations (i.e. transition below *d2* in Fig. 2). An invisible transition is created and connected to input and output places of each data operation that is allowed to be executed frequently. In this case, the input place of the invisible transition is an output place for that data operation while the output place of the invisible transition is an input place for that data operation (i.e. transition above *d1* in Fig. 2).

The first foundation of the synchronous product in our approach is *Model net*. The model net (N_M) is constructed by replacing each activity in the original process model (i.e. Fig. 1(a)) with corresponding operation net. Figure 2 shows the model net for our running example. In this model, we enriched the process model (Fig. 1(a)) with the expected data operations shown in Fig. 1(b).

The second foundation of the synchronous product in our approach is *Process net*. The process net (N_P) represents a process trace. It shows a sequence of the transitions labelled with activities and their life cycle as they appeared in the process trace.

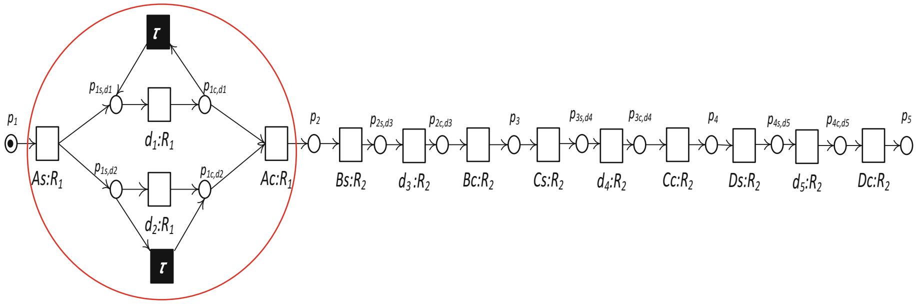


Fig. 2. Model net of the running example. The operation net of activity A is surrounded by the red line. (Color figure online)

The yellow part in the middle of Fig. 3 shows the process net constructed based on the process trace example in Fig. 1. Two concurrent transitions *Ac* And *Bs* in this model show the partial order of the completion of activity *A* (reflected in e_2) and the start of activity B (reflected in e_3) which have the same timestamp. To match start and complete events related to one instance of an activity, we consider a matching place labelled as C and the name of executed activity (we call these type of places as context places). The input and output of matching places are start and complete events related to one instance of an activity. It should be noted that context places are created if and only if the start and complete events related to one activity have the same “id” attribute.

The third foundation of the synchronous product in our approach is *Data net*. The data net (N_D) represents a data trace. It shows a sequence of the transitions labelled with executed data operations as they appeared in the data trace. The red part in the bottom of Fig. 3 shows the data net constructed based on the data trace example in Fig. 1.

Using the model net, process net and data net, we present the synchronous product model as the combination of these three nets with two additional sets of synchronous transitions. Figure 3 shows the synchronous product for our running example. For the sake of less complexity, in this model, we relabeled the transitions of model net as t_{mi}, transitions of process net as t_{pi}, and transitions in data net as t_{di}. We also chose new identifiers for the places in model net, process net and data net as p_{mi}, p_{pi} and p_{di}, respectively.

As shown in Fig. 3, other than transitions of the model, process and data nets, there are two sets of synchronous transitions called synchronous transitions and data synchronous transitions. *Synchronous transitions* only exist when an expected activity appears in the process net. *Data synchronous transitions* only exists when an expected data operation appears in the data net. Additionally, each data operation is associated to a so called matched activity. The matched activity is the activity instance that was executed by the same resource as the data operation and the timestamp of the data event should be between the start and completion time of the matched activity in the process net. These conditions are reflected in the model by input/output to the context place of matched activity. Input places of synchronous data operations contain: the input place

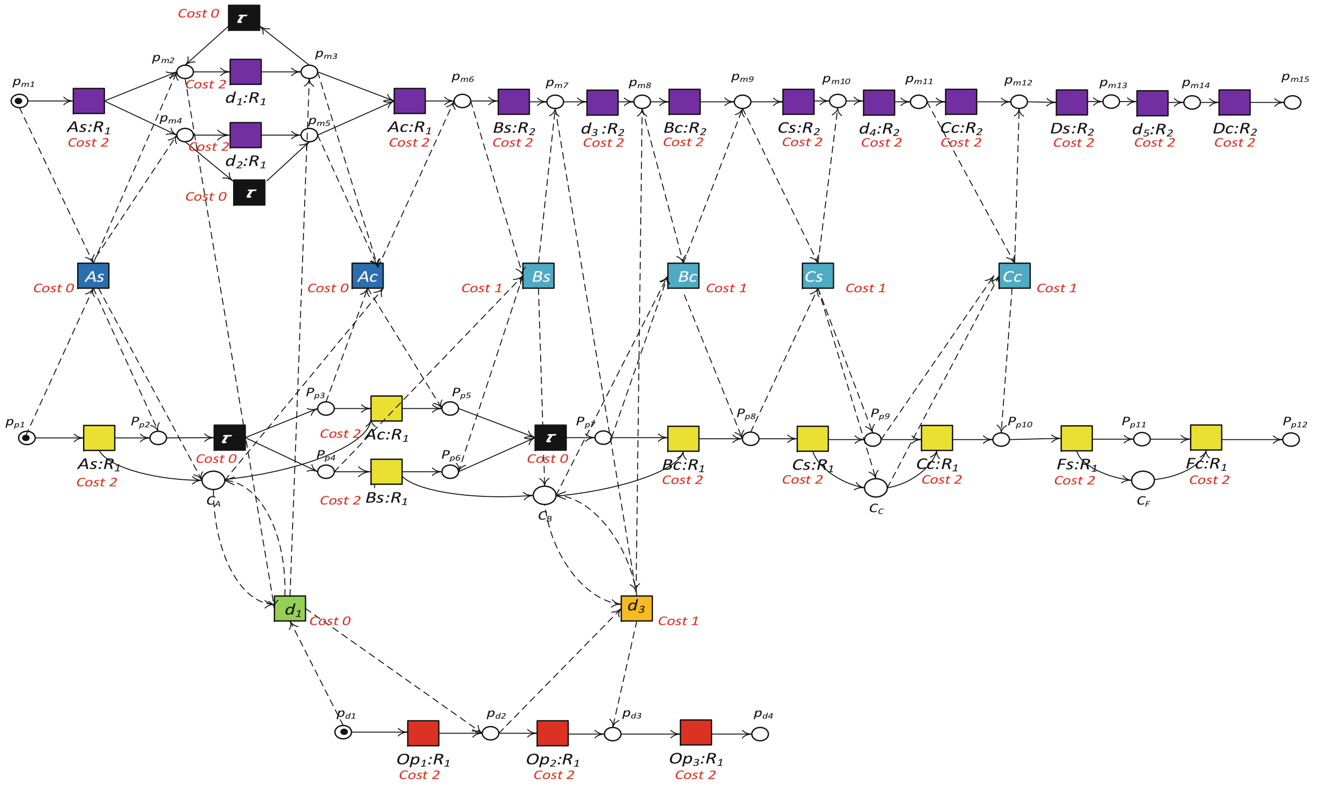


Fig. 3. Synchronous product model based on the inputs of the running example. The model net is depicted in the top part in purple color, the process net is depicted in the middle part in yellow color and the data net is depicted in the bottom part in red color. Synchronous moves and data synchronous moves are shown in blue and green colors, respectively. Synchronous moves and data synchronous moves with illegitimate roles are shown in light blue and orange colors, respectively (Color figure online).

of the corresponding executed data operation in the data net; the input place of the expected data operation in the model net; and the context place of matched activity in the process net. Output places of the synchronous data operations contain: the output places of the executed data operation; the output place of the expected data operation; and the context place of matched activity.

For including the privacy aspect in the synchronous transitions, we consider a penalty cost in case of expected activity and/or data operation done by an unexpected role. This will be discussed in the next section under the cost function definition.

2.2 Multi-layer Alignment and Cost Function

An alignment is a firing sequence of transitions from initial marking to the final marking in the synchronous product model. In our approach, initial marking m_i is the set of starting places of each model, process and data nets. Final marking m_f is the set of last places of each model, process and data nets. For instance, in Fig. 3, $m_i = \{p_{m1}, p_{p1}, p_{d1}\}$ is the initial marking and $m_f = \{p_{m15}, p_{p12}, p_{d4}\}$ is the final marking.

We need to relate "moves" in the logs to "moves" in the model in order to establish an alignment between the model, process trace and data trace. However, it might happen that some of the moves in the logs cannot be mimicked by the model and vice-versa. We explicitly denote such "no moves" by "$\gg$". Formally, we represent a move as (t_m, t_p, t_d), where we set t_m to be a transition in the model net, t_p to be a transition of the events in the process net (process trace), and t_d to be a transition of the events in data net (data trace). Our approach separates moves into two categories: synchronous moves and deviations. Synchronous moves represent expected behavior:

- A *synchronous move* happens when an expected activity was performed by a legitimate role.
- A *data synchronous move* happens when an expected data operation was executed by a legitimate role.

We further distinguish six kinds of deviations:

- A *move on model* happens when there are unobserved activity instances.
- A *move on model* happens when there are unobserved data operations.
- A *move on process log* happens when an unexpected activity instance was performed.
- A *move on data log* happens when an unexpected data operation was executed.
- A *synchronous move with illegitimate role* happens when an expected activity was performed by an illegitimate role.
- A *data synchronous move with illegitimate role* happens when an expected data operation was performed by an illegitimate role.

The computation of an optimal alignment relies on the definition of a proper cost function for the possible kinds of moves. We extend the standard cost function to include data and privacy costs. We define our default multi-layer alignment cost function as follows:

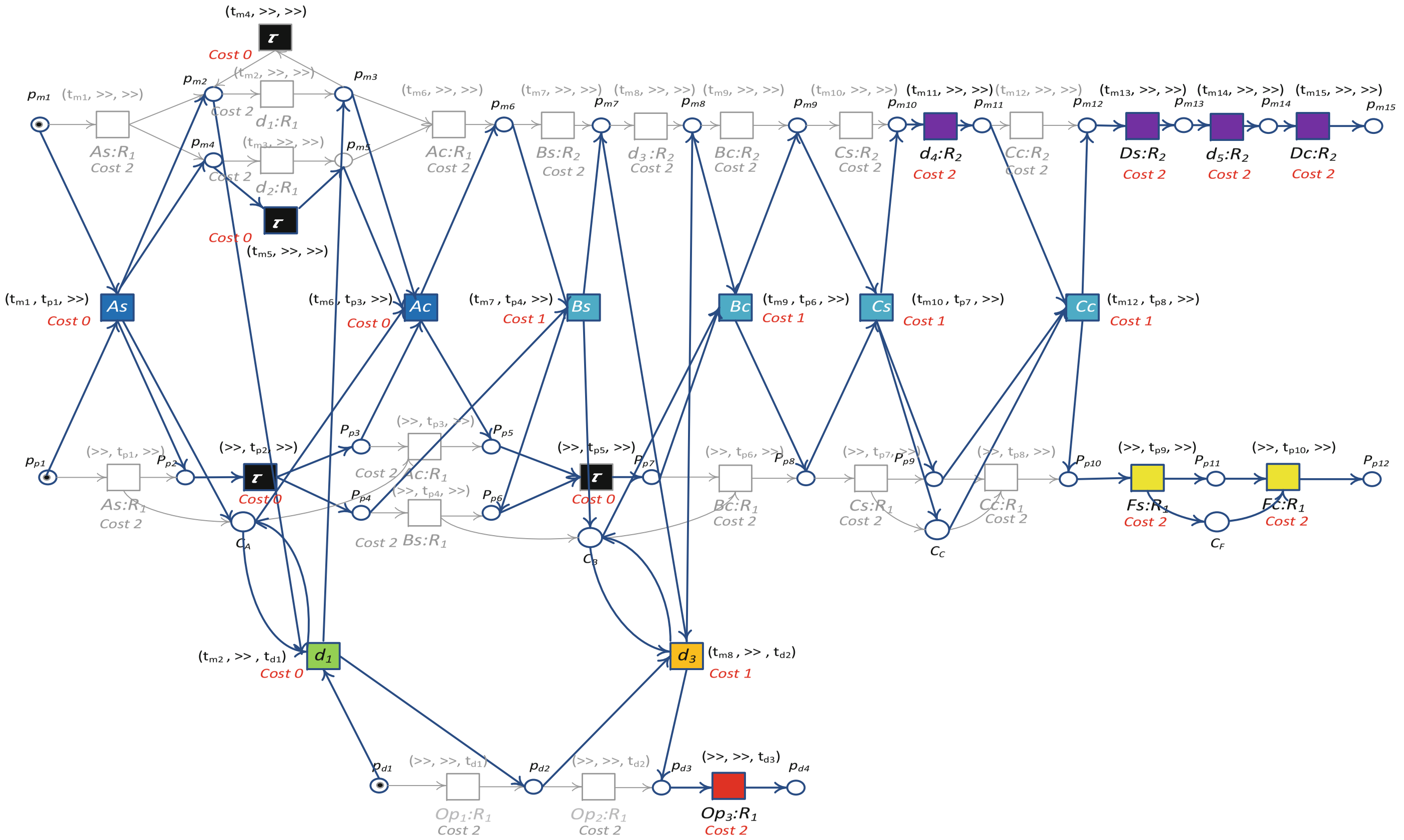


Fig. 4. Full run of the synchronous product corresponding to an optimal alignment, assuming a multi-layer cost function for the running example

Definition 1 (Multi-Layer Alignment Cost function). *Let (t_m, t_p, t_d) be a move in alignment between a model, process trace and a data trace. The cost $K(t_m, t_p, t_d)$ is:*

$$K(t_m, t_p, t_d) = \begin{cases} 2, & \textit{if } (t_m, t_p, t_d) \textit{ is a move on process log} \\ & \textit{or move on data log, or move on model} \\ 0, & \textit{if } (t_m, t_p, t_d) \textit{ is Process/Data sync. move} \\ & \textit{with legitimate role} \\ 1, & \textit{if } (t_m, t_p, t_d) \textit{ is process/Data sync. move} \\ & \textit{with illegitimate role} \end{cases}$$

Note that, to include the cost for deviations related to the privacy layer, we considered a penalty cost equal to 1 in our cost function. If the actor of observed behavior was not allowed to perform activity and/or data operation we add the penalty cost.

The alignment with the lowest cost is called an optimal alignment. We define Optimal Multi-Layer Alignment as follows:

Definition 2 (Optimal Multi-Layer Alignment). *Let N be a WFR-net, σ_c and β_c be a process trace and data trace, respectively. Assuming $\mathcal{A}_\mathcal{N}$ as the set of all legal alignment moves, a cost function K assigns a non-negative cost to each legal move: $\mathcal{A}_\mathcal{N} \rightarrow \mathbb{R}_0^+$. The cost of an alignment γ between σ_c, β_c and N is computed as the sum of the cost of all constituent moves $\mathcal{K}(\gamma) = \sum_{(t_m, t_p, t_d) \in \gamma} K(t_m, t_p, t_d)$. Alignment γ is an optimal alignment if for any alignment γ' of σ_c, β_c and N, $\mathcal{K}(\gamma) \leq \mathcal{K}(\gamma')$.*

For finding the optimal alignments we employed A* algorithm. Figure 4 illustrates an optimal alignment for running example, depicted on top of the synchronous product shown in Fig. 3. It shows that there are six kinds of deviations between observed behavior and modeled behavior, namely synchronous moves with illegitimate roles on transitions *Bs*, *Bc*, *Cs*, and *Cc* in light blue color, data synchronous move with illegitimate role showing spurious data operation on transitions *d3* in orange color, model moves showing missing data operations on transitions *d4* and *d5* and model moves showing skipped activities on transitions *Ds* and *Dc* in purple color, process log moves indicating unexpected activities on transitions *Fs* and *Fc* in yellow color, and a data log move showing unexpected data operations on transition *op3* in red color.

3 Evaluation

To evaluate the applicability of our approach to real-life scenarios, we used the event log recording the loan management process of a Dutch Financial Institute provided by BPI challenge 2017 [5]. After splitting the provided event log, the resulting process log and data log contain 301,709 workflow events and 256,767 data operations, respectively. These logs were recorded from managing 26,053 loan applications. The activities and data operations were performed by 146 resources (employees or system).

Figure 5 shows the loan management process in Petri net notation. In this process, there are four main milestones: receiving applications, negotiating offers, validating documents, and detecting potential fraud. The execution of activities may require performing certain mandatory or optional data operations. The data model of this process which presents the relationship between activities and data operations is shown in

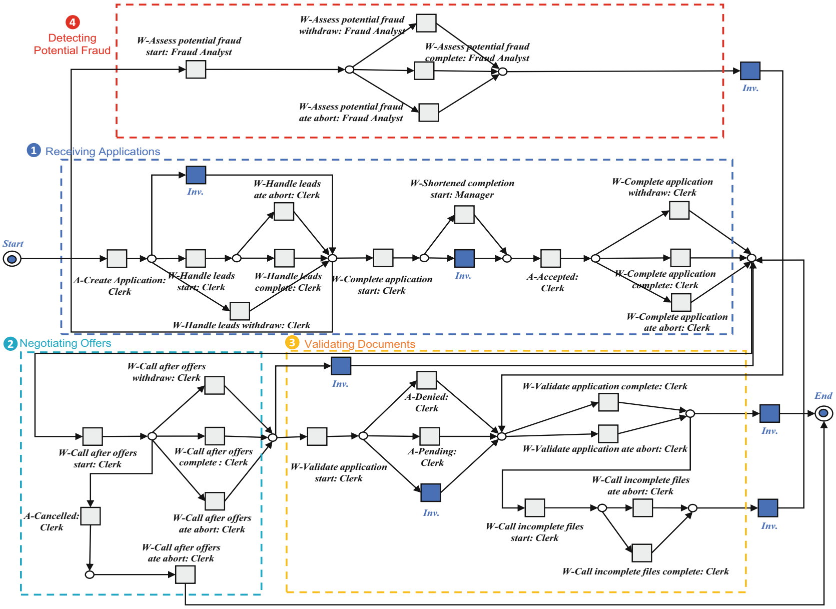


Fig. 5. Loan management process model [10]

Table 1. Such data model is created according to domain knowledge and also indicates whether the user is allowed to repeat the execution of the data operations. As shown in the process model (Fig. 5), three roles are supposed to conduct the activities. Most of the activities are supposed to be done by the role *clerk*. Activities related to fraud detection are supposed to be done by a *fraud analyst*. The activity "W Shortened completion"

Table 1. Data model of the loan management process. Type: Mandatory (M), Optional (O). Repetition: is allowed (True), is not allowed (False). A: Application, O: Offer, W: Workflow [10].

Activity	*Data operation*	*Type*	*Repetition*
A-Create Application	Create: (applicationID)	M	False
W-Shortened Completion start	Read: (applicationID, email)	M	False
A-Accepted	Create: (offerID)	M	False
	Read: (offerID)	M	False
	Read: (address, email)	M	False
	Read: (address)	O	False
A-Cancelled	Update: (OCancelledFlag)	M	True
W-Call after offer start	Update: (ACompletedFlag)	M	False
W-Call after offer complete	Update: (OAcceptedFlag)	M	False
W-Call after offer withdraw	Update: (OReturnedFlag)	M	False
W-Call after offer ate abort	Update: (OCancelledFlag)	M	False
W-Validate application start	Update: (AValidatedFlag)	M	False
A-Pending	Update: (OAcceptedFlag)	M	False
A-Denied	Update: (ORefusedFlag)	M	True
W-Validate application ate abort	Update: (OCancelledFlag)	M	True
W-Call incomplete files start	Update: (AInCompleteFlag)	M	False

Table 2. The result of experiment with real-life data

Category	*Anomaly*	*Occurence*
1	Ignored mandatory data operation "Update(OCancelledFlag)"	40,869
2	Unexpected data operation "Update (OReturnedFlag)"	18,291
3	Unexpected data operation "Read (offID)"	11,874
4	Unexpected data operation "Create (offID)"	11,874
5	Unexpected data operation "Read (address- email)"	9,640
6	Ignored mandatory data operation "Create(appID)"	9,354
7	Unexpected data operation "Update (OCancelledFlag)"	5,624
8	Unexpected activity "W Call incomplete files complete"	2,565
9	Skipped activity "W Call incomplete files ate abort"	1,919
10	Unexpected data operation "Read (email)"	1,874

can only be executed by a *manager*. Managers also have the authority to perform all the activities related to a clerk.

We implemented our approach as a package in the ProM framework called Multi Layer Alignment in the "MultiLayerAlignmentWithContext" plugin. Using this tool, we applied our approach on the described business process. A summary of our results that shows ten most frequent anomalies is reported in Table 2. In addition to detecting multi-layer deviations, the experiment remarks that the approach is capable to reconstruct and provide the link between performed activities in the process layer and executed data operations in the data layer to present the contexts of data processing. For example, Table 2 shows mandatory data operation "*Update(OCancelledFlag)*" was ignored 40,869 times. We have also developed a view that provides detailed information, described in [10], which finds that this anomaly happened 16,735 times in the context of activity "*W-Validate application ate abort*", 16,184 times in the context of activity "*W-Call after offers ate abort*", and 7,950 times in the context of activity "*A-Cancelled*". Furthermore, it could detect who (in terms of roles and users) had the anomalous or suspicious behaviors during process executions.

4 Conclusion

In this work, we presented an approach for detecting complex anomalous behaviors in business processes. Through an example, we showed the structure of our multi-layer synchronous product model which is the foundation of conformance checking and applying alignment algorithms.

In existing multi-perspective conformance checking approaches, control flow perspective is a priority thus many deviations stay hidden and uncovered. In contrast, in our approach, different perspectives of a business process such as control flow, data and privacy aspects are considered simultaneously to detect complex anomalies which relates to multiple perspectives of a business process.

We showed the applicability of our approach using real-life event logs of a loan management process from a financial institute. The experiment demonstrated the approach's capability to return anomalies such as ignored data operations, suspicious activities and data operations, spurious and unexpected data operations. Additionally, our method could reconstruct the link between process layer and data layer from executed behavior and present the contexts of data processing. Thus, it can discover data accesses without clear context and purposes.

As future step, we plan a qualitative analysis of how useful the results of our approach are to the business analysts to detect anomalous and suspicious behaviors in business processes.

Reproducibility. The inputs required to reproduce the experiments can be found at https://github.com/AzadehMozafariMehr/Multi-PerspectiveConformanceChecking

Acknowledgement. The author has received funding within the BPR4GDPR project from the European Union's Horizon 2020 research and innovation programme under grant agreement No 787149.

References

1. Van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisc. Rev. Data Min. Knowl. Discov. **2**(2), 182–192 (2012)
2. Alizadeh, M., Lu, X., Fahland, D., Zannone, N., van der Aalst, W.: Linking data and process perspectives for conformance analysis. Comput. Secur. **73**, 172–193 (2018)
3. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking Relating Processes and Models. Springer, Cham (2018)
4. Chandola, V., Banerjee, A., Kumar, V.: Anomaly Detection. In: Sammut, C., Webb, G. (eds.) Encyclopedia of Machine Learning and Data Mining, pp. 1–15. Springer, Boston (2016). https://doi.org/10.1007/978-1-4899-7502-7_912-1
5. van Dongen, B.: BPI Challenge 2017 (2017). https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b
6. 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.) Business Process Management. LNCS, pp. 113–129. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40176-3_10
7. Lu, X., Fahland, D., van der Aalst, W.M.P.: Conformance checking based on partially ordered event data. In: Fournier, F., Mendling, J. (eds.) BPM 2014. LNBIP, pp. 75–88. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15895-2_7
8. Mannhardt, F., Leoni, de, M., Reijers, H., Aalst, van der, W.: Balanced multi-perspective checking of process conformance. Computing **98**, 407–437 (2014)
9. Mozafari Mehr, A.S., de Carvalho, R.M., van Dongen, B.: Detecting privacy, data and control-flow deviations in business processes. In: Nurcan, S., Korthaus, A. (eds.) CAiSE 2021. LNBIP, pp. 82–91. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-79108-7_10
10. Mozafari Mehr, A.S., de Carvalho, R.M., van Dongen, B.: An association rule mining-based framework for the discovery of anomalous behavioral patterns. In: Chen, W., Yao, L., Cai, T., Pan, S., Shen, T., Li, X. (eds.) ADMA 2022. LNCS, vol. 13725, pp. 397–412. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22064-7_29

11. Nolle, T., Seeliger, A., Mühlhäuser, M.: BINet: multivariate business process anomaly detection using deep learning. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNCS, vol. 11080, pp. 271–287. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_16
12. Dongen, B.F.: Efficiently computing alignments. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNCS, vol. 11080, pp. 197–214. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_12