# Attribute-Based Conformance Diagnosis: Correlating Trace Attributes with Process Conformance

Michael Grohs[1 [0000−0003−2658−8992]] and Jana-Rebecca Rehse[1 [0000−0001−5707−6944]]

University of Mannheim, Mannheim, Germany
{mgrohs@mail.,rehse@}uni-mannheim.de

**Abstract.** An important practical capability of conformance checking is that organizations can use it to alleviate potential deviations from the intended process behavior. However, existing techniques only identify these deviations, but do not provide insights on potential explanations, which could help to improve the process. In this paper, we present attribute-based conformance diagnosis (ABCD), a novel approach for correlating process conformance with trace attributes. ABCD builds on existing conformance checking techniques and uses machine learning techniques to find trace attribute values that potentially impact the process conformance. It creates a regression tree to identify those attribute combinations that correlate with higher or lower trace fitness. We evaluate the explanatory power, computational efficiency, and generated insights of ABCD based on publicly available event logs. The evaluation shows that ABCD can find correlations of trace attribute combinations with higher or lower fitness in a sufficiently efficient way, although computation time increases for larger log sizes.

**Keywords:** Process Mining · Conformance Checking · Correlations · Trace Attributes · Root Cause Analysis.

## 1 Introduction

The goal of conformance checking is to analyze the relation between the intended behavior of a process, captured in a process model, and the observed behavior of a process, captured in an event log [7]. It generates insights on where and how the observed behavior aligns with or deviates from the intended behavior. Organizations can use these insights for example to check whether their process execution is compliant with the originally designed process [22]. Over the last years, multiple conformance checking techniques have been developed, including rule checking, token-based replay, and alignments [7]. The techniques differ with regards to their algorithmic approach, computational complexity, and generated results, but they have one output in common: A measure of the conformance between log and model, called fitness, which quantifies the capability of a model to replay the behavior observed in the log [22].

One problem of existing conformance checking techniques is that they do not enable practitioners to reach their underlying goal, which is to improve the process [19]. As an example, consider a loan application process in a bank, where the application of a conformance checking algorithm yielded an overall fitness value of 0.8. From this number, a process analyst can conclude that some deviations between log and model occurred, but they do not know where, how, and—most importantly—why the process execution deviated and what the effects of the potential problem are. Therefore, explaining and understanding the underlying causes of conformance problems is an important part of leveraging the practical benefits of conformance checking [22]. Existing conformance checking techniques focus only on the identification of deviations and do not provide any potential reasons for their occurrence [5], although this would be a vital prerequisite for any deeper process analysis. For our exemplary loan application process, if the process analyst knows that loans with a higher amount more likely deviate from the intended process, they could specifically analyze those process instances to find and eventually address the root cause of those deviations.

In this paper, we present a novel approach for finding correlations between process conformance and trace attributes. This approach, called attribute-based conformance diagnosis (ABCD), builds on the results of existing conformance checking techniques and uses machine learning to find trace attribute values that potentially impact the conformance. Specifically, it creates a regression tree to identify those attribute combinations that correlate with higher or lower trace fitness. These correlations can be considered as potential explanations for conformance differences and therefore as a starting point for further analysis steps to find and address the causes of lower process conformance. ABCD is (1) inductive, i.e., it requires no additional domain or process knowledge, (2) data-driven, i.e., it requires only an event log and a process model as input, (3) universally applicable, i.e., it does not depend on process-specific characteristics, and (4) flexible, i.e., it can be configured to fit a specific case.

In the following, the ABCD approach is introduced in Sect. 2. Its explanatory power, computational efficiency, and potential practical insights are evaluated based on publicly available event logs in Sect. 3. We discuss related work in Sect. 4 and conclude with a discussion of limitations and future work in Sect. 5.

## 2 Approach

The goal of the ABCD approach is to find attribute value combinations in an event log that correlate with differences in conformance. Therefore, it analyzes trace attributes and correlates them with trace-level fitness, which is the most common way to measure conformance [22]. A schematic overview of ABCD can be found in Fig. 1. The approach requires two inputs, an event log and a corresponding process model, and consists of two major steps. In the first step, explained in Sect. 2.1, we enrich the event log with the trace-level fitness values with regard to the provided process model. This enriched log serves as input for the second step, called Inductive Overall Analysis (IOA) and explained in

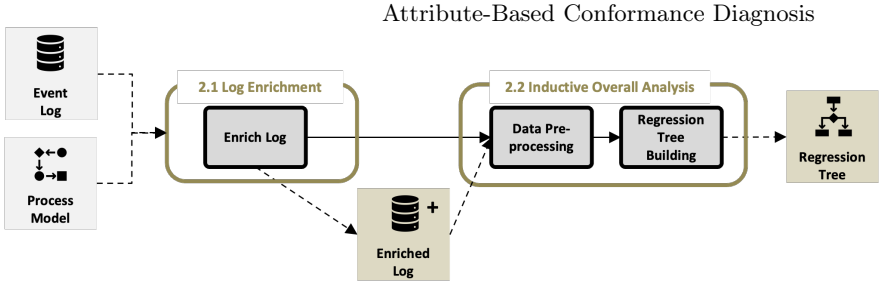


**Fig. 1.** Illustration of the Attribute-Based Conformance Diagnosis (ABCD) Approach

Sect. 2.2. It determines the correlations between combinations of attribute values and process conformance. Therefore, it computes a regression tree. Regression trees are a data mining technique that relate a set of independent variables, in our case all trace attributes in an event log, to a real-valued dependent variable, in our case , i.e., average trace fitness in a log. To build the regression tree, the event log is iteratively split into sub-logs, based on trace attribute values. Each split defines a new node in the tree. These nodes are then used to predict the value of the dependent variable [10]. To find the best fitting tree, the algorithm minimizes the sum of errors in the prediction. An error is the difference between the predicted value in a leaf node and the actual value of the respective sub-log. The percentage of the true variation that can be explained by the predictions, i.e., 1 minus the sum of errors, is the coefficient of determination $R^2$, which can be used to determine the prediction quality of the regression tree [9].

### 2.1 Log Enrichment

Because the goal of ABCD is to correlate trace attributes with variations in conformance, it needs the trace-level fitness to perform any further analysis. Therefore, we compute the fitness of each trace with regard to the provided process model and add the value to the event log as a trace attribute. The user can choose between token-replay fitness and alignment-based fitness [7]. The latter is the default choice used in the remainder of this paper. This parametrization allows users to flexibly choose the best-suited technique, for example choosing token-based fitness if alignments require too much computation time.

After computing the trace fitness value, we also enrich each trace by its overall duration, defined as the time difference between start and end event in a timely ordered trace. This ensures that at least one trace attribute will always occur in the log. We decided on the trace duration as the default trace attribute, because it can be computed for every (time-stamped) event log and because the relation between process performance and process conformance is potentially relevant for all processes, independent of their context [24].

### 2.2 Inductive Overall Analysis

Following the log enrichment, Inductive Overall Analysis (IOA) determines correlations between combinations of attribute values and process conformance.

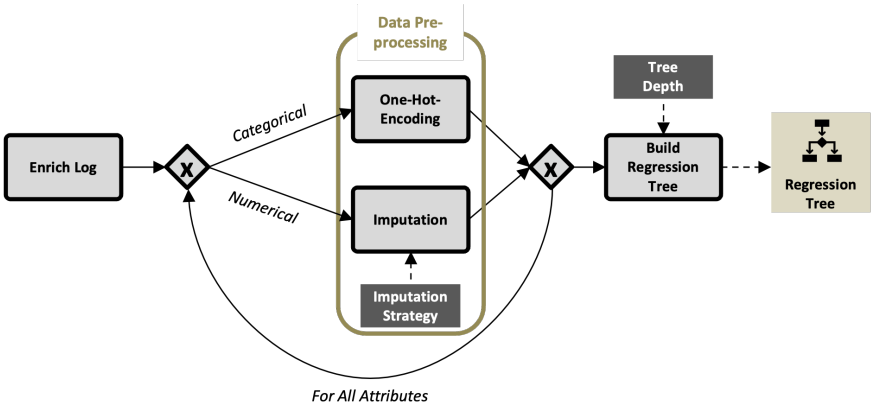


**Fig. 2.** Illustration of Inductive Overall Analysis (IOA)

Therefore, it first preprocesses the data and then constructs a regression tree that uses the trace attribute values as determinants for the fitness value. Fig. 2 shows the schematic overview. IOA consists of two steps: preprocessing the data and building the regression tree.

**Data Preprocessing.** For the data pre-processing, we distinguish between categorical and numerical attributes. Due to requirements of the tree algorithm, pre-processing is necessary for both. First, because a regression tree can only handle numerical attributes, categorical variables need to be encoded to be used as a determinant. For this purpose, we use One-Hot-Encoding, which constructs one binary trace attribute per categorical attribute value. Second, the regression tree algorithm cannot handle missing data. If there are values missing for numerical attributes, we need to perform imputation, i.e., replace missing values with other values [31]. Assuming that raw data is the best representation of reality, no imputation will be the default. If it must be performed due to missing values, potential imputation strategies include replacing missing values with the mean, the median, the most frequent value, or a constant value. For IOA, users can select the imputation strategy as a parameter. Additional to no imputation, we allow for imputing with the most frequent value, a constant value of 0, the mean, and the median value. Imputation will only be necessary for numerical attributes since the encoding transforms the categorical attributes into binary attributes with no missing values. Missing values in categorical attributes will therefore lead to a 0 in all binary attributes.

**Regression Tree Building.** After the preprocessing, we build the regression tree. The goal is to find those combinations of attribute values that best predict variations in conformance. Therefore, the regression tree consists of nodes that split the event log based on one attribute value. A splitting node includes a condition for the attribute value, e.g., a duration smaller than 4 days. For all traces below the splitting node on the left side of the tree, the node condition is true. For all traces below on the right side, it is false. Leaf nodes do not state a condition, either because the tree has reached its maximum depth or because an additional split will not improve the result. Traversing the tree from root to

leaves, each node divides the log according to its condition, iteratively dividing the log into one sub-log per leaf node. The sub-log of an internal node is the union of all sub-logs of its children. Each node reports on the average fitness for the sub-log created by all splits above it, which is used as a predictor for the fitness of the individual traces. The tree algorithm chooses attribute values and conditions by minimizing the total errors in the prediction, i.e., the sum of the differences between the true fitness value of each trace and the average fitness in the leaf node. The final tree consists of splitting nodes and leaf nodes. The leaf nodes indicate the overall prediction for the sub-logs created by the splitting nodes. The combination of conditions leading down to a leaf node indicates a combination of attribute values that well predicts the fitness of the given sub-log, i.e., it consistently determines the conformance level of these traces.

For building the tree, we use the sklearn-environment in Python[1]. As a parameter, we require the maximum tree depth, i.e., the number of node layers the algorithm may use to split the log. When choosing this depth, we need to balance the explanatory power of the tree with its visual clarity and the granularity of sub-logs. The returned regression tree includes those attribute value combinations that are correlated with higher or lower fitness and thus offer a potential explanation for differences in conformance.

## 3 Evaluation

We implemented the ABCD approach in Python.[2] Using this implementation, we conduct an evaluation to show that ABCD has explanatory power, is computationally efficient, and generates practical insights. For our evaluation, we used three publicly available data sets consisting of seven event logs (see Tab. 1):

**MobIS-Challenge 2019 [26].** This event log from a travel management process contains trace attributes. It also comes with a matching process model that describes that process and can be used as a reference for conformance checking.

**BPI Challenge (BPIC) 2020 [30].** This collection of five event logs, also from a travel management process, contains many trace attributes, which makes it well suitable to test ABCD's abilities to provide insights. Because there is no to-be model available for this process, we applied the PM4Py auto-filter on the event log to filter all common variants[3] and discovered a model using the Inductive Miner. This way, we check conformance against the most frequent behavior.

**BPI Challenge (BPIC) 2017 [29].** This event log from a loan application process is comparably large, which makes it well suitable to test ABCD's computational feasibility. Because there also is no to-be model available for this process, we discovered one using the above-described method.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html`

[2] `https://gitlab.uni-mannheim.de/mgrohs/attribute-based-conformance-diagnosis/-/tree/main`

[3] `https://pm4py.fit.fraunhofer.de/documentation`

**Table 1.** Public Event Logs used for Evaluation

| ID | Dataset | Name | Traces | Events | Trace attributes |
|---|---|---|---|---|---|
| (1) | MobIS | MobIS | 6,555 | 83,256 | Duration (Dur), Costs |
| (2) | BPIC 2020 | Domestic Declarations | 10,500 | 56,437 | Dur, Amount (Amn), Budget Number (BudNo), Declaration Number (DeclNo) |
| (3) | BPIC 2020 | International Declarations | 6,449 | 72,151 | Dur, Adjusted Amn, Amn, BudNo, DeclNo, Original Amn, Act. No., Org. Entity, Req. Bud. |
| (4) | BPIC 2020 | Request for Payment | 6,886 | 36,796 | Dur, Act., Cost Type, Org. Entity, Project, Req. Amon., Task, Rfp No. |
| (5) | BPIC 2020 | Prepaid Travel Costs | 2,099 | 18,246 | Dur., Act., Cost Type, Org. Entity, Project No., Bud. No., Red. Budget, Project, Task |
| (6) | BPIC 2020 | Travel Permit | 7,065 | 86,581 | Dur., Bud. No., Cost Type, Org. Entity, Overspent Amn, Project, Req. Amn |
| (7) | BPIC 2017 | Loan Application | 31,509 | 1,202,267 | Dur, Application Type, Loan Goal, Requested Amn |

### 3.1 Explanatory Power

To measure the explanatory power of ABCD, we use the coefficient of determination $R^2$, which shows the goodness of fit of the regression [8]. To determine the influence of our parameters, our evaluation setting varies the imputation strategy (none, mean, median, zero, constant), and the tree depth (from 3 to 7; a larger tree would not be visually clear anymore).

We first inspect the influence of the imputation strategy. This is shown in Tab. 2, where we list the $R^2$ for the four imputation strategies for the MobIS event log. No imputation is not possible for this event log due to missing attribute values. We do see not see any difference in $R^2$ for the different imputation strategies in the MobIS data. This is also the case for all other logs.[4] We can conclude that the imputation strategy has no effect on the explanatory power of ABCD. However, this might be different for highly variable real-life event logs, so the imputation option is necessary to remain universally applicable.

Tab. 3 shows the $R^2$ values for all event logs and tree depths. As expected, $R^2$ grows with tree depth, due to more allowed splits in the tree. This increase is log-dependent and ranges between 1% for log (2) and 13% for log (3). It is generally impossible to determine a universal threshold for a good $R^2$ value [25]. However, we see that ABCD is capable of explaining at least one fifth of the fitness variation in all logs and as much as 84% in one, meaning that it is capable of finding correlations of data attributes with (non-)conformance. All in all, our evaluation showed that for our inspected datasets, ABCD has moderate to high explanatory power and is not sensitive to imputation and tree depth.

[4] The full evaluation documentation is available in the GitLab repository

**Table 2.** $R^2$ for the MobIS Data Set for Working Imputation Strategies and Tree Depths 3 to 7

| Log | Imp. | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| (1) | mean | 0.163 | 0.178 | 0.195 | 0.203 | 0.218 |
| (1) | median | 0.163 | 0.178 | 0.195 | 0.203 | 0.218 |
| (1) | zero | 0.163 | 0.178 | 0.195 | 0.203 | 0.218 |
| (1) | freq. | 0.163 | 0.178 | 0.195 | 0.203 | 0.218 |

**Table 3.** $R^2$ for Tree Depths 3 to 7

| Log | Imp. | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| (1) | all | 0.163 | 0.178 | 0.195 | 0.203 | 0.218 |
| (2) | all | 0.744 | 0.747 | 0.75 | 0.753 | 0.756 |
| (3) | all | 0.303 | 0.344 | 0.374 | 0.405 | 0.433 |
| (4) | all | 0.819 | 0.824 | 0.829 | 0.835 | 0.84 |
| (5) | all | 0.483 | 0.513 | 0.54 | 0.569 | 0.596 |
| (6) | all | 0.416 | 0.434 | 0.448 | 0.462 | 0.476 |
| (7) | all | 0.322 | 0.336 | 0.348 | 0.357 | 0.368 |

### 3.2 Computational Efficiency

For assessing the computational efficiency of ABCD, we measure the execution times, separated into the enrichment step in Tab. 4 and the analysis step in Tab. 5. Each reported value in those tables is an average of three separate executions, to account for outliers. For the analysis time, we only report the average execution time over all imputation strategies since there were no significant deviations between them.

**Table 4.** Enrichment Times

| Log | Traces | Events | Attr. | Time [s] |
|---|---|---|---|---|
| (1) | 6,555 | 83,256 | 2 | 33.75 |
| (2) | 10,500 | 56,437 | 5 | 2.26 |
| (3) | 6,449 | 72,151 | 18 | 61.98 |
| (4) | 6,886 | 36,796 | 9 | 1.38 |
| (5) | 2,099 | 18,246 | 17 | 3.65 |
| (6) | 7,065 | 86,581 | 168 | 859.47 |
| (7) | 31,509 | 1,202,267 | 4 | 9,216.35 |

We see that the enrichment time increases with the number of traces and the number of events, because especially alignments become computationally expensive [7]. Additionally, the number of trace attributes negatively influences the enrichment time, which is visible for the Travel Permit log (6). At most, the enrichment takes 2.6 hours for the largest log (7).

**Table 5.** Average Computation Time for IOA over All Imputation Strategies for Tree Depths 3 to 7 in s

| Log | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|
| (1) | 64.61 | 65.13 | 64.98 | 65.69 | 64.77 |
| (2) | 180.99 | 180.02 | 181.69 | 183.07 | 187.44 |
| (3) | 132.15 | 132.71 | 132.45 | 134.88 | 132.98 |
| (4) | 99.51 | 100.42 | 99.78 | 100.14 | 101.1 |
| (5) | 15.95 | 16.03 | 16.0 | 16.19 | 16.33 |
| (6) | 602.4 | 618.54 | 591.85 | 590.82 | 651.73 |
| (7) | 1,395.65 | 1,357.93 | 1,353.86 | 1,352.12 | 1,375.79 |

Like the enrichment time, the analysis time for IOA depends heavily on the number of traces and the number of trace attributes, again visible for logs (6) and (7). However, this increase is less significant compared to the increase in enrichment time and the maximum duration is below 25 minutes. In case of more trace attributes, we consider more independent variables and in case of many traces we have a larger sample size, both increasing the explanatory power of ABCD. We conclude that ABCD is computationally feasible even for larger logs, although the execution times are a potential drawback. Neither imputation strategy nor tree depth have a significant impact on the analysis time.

Overall, we see a negative influence of the log size on the computational efficiency. Still, execution takes less than 3 hours for event logs with up to 1.2 million events. Considering the potential value of ABCD, the execution time does not limit its applicability. As alignment are the main cause for long executions, larger event logs could still be analyzed by means of a different fitness technique.

### 3.3 Practical Insights

The main benefit of ABCD is that it generates process insights without prior knowledge, which is supposed to provide value for practitioners. These insights are correlations between trace attributes and process conformance that serve as a starting point for further process analyses. To demonstrate some of these insights, we further examine the regression trees generated for the event logs. It is important to note that for all event logs except MobIS, the process model is generated based on variant filters. This means that conformance and fitness are based on the most common variants and not on a constructed process model. In the following, conformance of the BPI logs has to be interpreted as conformance to the most common variants. Detailed information about the practical insights provided by ABCD can be derived from the computed regression trees for all logs (available in the GitLab repository).

**MobIS.** An exemplary regression tree with depth 3 is provided in Fig. 3. It splits the log into six different sub-logs represented by the six leaf nodes. For example, the top node splits the log based on whether the trace has a duration above 0 (more than one event). The color indicates the fitness value: high fitness leads to darker color. We see that short duration above 0 correlates with better conformance. For traces with one event, lower costs correlate with slightly better fitness.

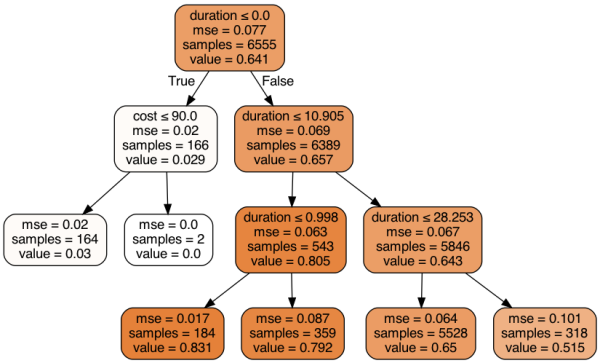


**Fig. 3.** Exemplary Regression Tree for the MobIS Log

**BPI Challenge 2020.** Not knowing the trace ID, e.g., the declaration number, correlates with lower conformance in logs (2), (3), (4), and (5). For all five logs, the duration is an important feature in the trees, which shows the value of separately enriching this attribute. Longer traces conform better in log (2), but they conform worse in log (4). Another relevant trace attribute is the requested amount or budget, which also correlates with lower conformance in most cases.

**BPI Challenge 2017.** Longer traces conform better for log (7). Further, an unknown loan goal and a smaller requested amount correlate with lower fitness.

We conclude that ABCD can generate practical insights in form of correlations between trace attributes and trace fitness without relying on process or domain knowledge. These correlations can serve as starting points to identify causalities that explain conformance deviations. We show that it finds significant attribute values correlating with worse conformance, both for available to-be models and for mined models that represent the most common behavior. The identified correlations can be used to further examine the deviations that occur in the sub-logs created by the regression tree nodes. Comparing all sub-logs of MobIS data based on the leaf nodes in Fig. 3 could yield additional insights into conformance variation, including, e.g., the location and type of deviation that occurs in the individual sub-logs. For example, we see that for the leaf node with size 184, the deviations occur primarily in the reporting part of the travel management process.

## 4 Related Work

In this section, we elaborate on work related to the ABCD approach. Many other approaches combine data attributes and conformance checking. For example, data attributes are used while performing the conformance check to incorporate other perspectives into the optimal alignment of data-enriched process models and event logs [22,20,21]. Data attributes can also be used to define response moves (i.e., log moves that change data attributes that have been incorrectly changed by another log move in advance) [28] and to perform multi-perspective conformance checks on declarative models [6]. In all approaches, the data attributes refine the check itself but are not used to potentially explain conformance problems.

Data attributes can also be used to create sub-logs or sub-models in so called process cubes. Users can then analyze the differences between the sub-logs or sub-models and draw conclusions about what data attributes lead to the differences [1]. Main applications are process discovery [14,17] and performance analysis [4,2]. Applying process cubes for various purposes implicitly tries to use data attributes to explain differences in an event log or process model, often related to performance. This resembles attribute-based conformance diagnosis, but focuses on aspects other than conformance and metrics other than fitness.

The research stream that resembles ABCD the most closely is called root cause analysis (RCA). It aims to identify causal structures between different variables and show the influence these variables have on each other [23]. This can be achieved by using structural equation models based on data attributes [23], Granger-causal feature pairs, conventional correlations [3,18], or clustering techniques [12]. Also, to find reasons for deviations in processes, fuzzy mining and rule mining with data attributes can be applied without performing any conformance check [27]. Consequently, no deviations against a to-be model are investigated.

Another prominently used RCA technique are regression trees [10,16]. In process mining, regression trees have been applied to detect causes for

performance issues [16], for example by analyzing data attributes not referring to the control-flow [10]. Also, tree structures can be applied to identify causes for control-flow deviations located through sub-group discovery [11]. However, all approaches require domain knowledge to identify deviations or validate root causes after the automated analysis. Further, current approaches do not use conformance as the dependent variable. The automation is limited and the approaches are very specific [10].

Correlation-based RCA is also supported by process mining tools like Appian Process Mining, ARIS PM, Celonis, Lana Labs and Mehrwerk Process Mining. Those tools among others have been identified as relevant in a recent study [15]. However, none of them include a to-be model in the analysis but try to find root causes for variations in the data instead variations in conformance.

ABCD further resembles approaches like [11,12] where correlations between data attributes and process flow metrics other than conformance are identified. However, no to-be models are included in the analysis and therefore no conformance checking can be performed.

## 5 Discussion & Conclusion

The goal of the ABCD approach is to identify combinations of trace attribute values that correlate with variations in process conformance. Therefore, we first enrich the event log with fitness values. After that, we investigate the correlation between process conformance and attribute combinations. Our evaluation shows that ABCD is able to generate practical insights with explanatory power in an acceptable computation time. ABCD is inductive because it does not rely on domain knowledge and data-driven because it only needs an event log and a corresponding process model. It is universally applicable because is only depends on generic event log attributes, such as timestamps, and flexible because users can parametrize it to fit their specific case.

ABCD is subject to multiple limitations, which should be addressed in future research. First and most importantly, ABCD identifies correlations between attribute values and process conformance. It is not capable to determine whether and how the identified values actually caused the process to deviate. Instead, they are meant as an orientation for practitioners that try to improve the conformance of their process. In future research, ABCD could be extended by causal analysis techniques that are capable of identifying causal relations between attribute values and process conformance. Currently, the causal identification is performed manually based on the found correlations (i.e., potential explanations).

Second, the computation times indicate that the enrichment might take long for larger event logs, mainly due to the duration of the alignments. To still make ABCD applicable to larger event logs, we could compute the trace fitness with other techniques such as token-based replay or heuristics [7]. This was not necessary for our evaluation, because the duration of under three hours at maximum was acceptable, but it might become necessary for larger data sets.

Third, we enriched traces by their duration only. This attribute was useful since the case study found it to be a potential explanatory factor in many regression trees. However, additional enrichment by other generic trace attributes might further increase the explanatory power. Possibilities are the weekday in which the trace started or the number of other active cases at the point of initiation. Such attributes could also relate to events, such as the occurrence of certain activities in a trace or the number of executions of the same activity. More sophisticated encoding approaches might be used [13].

Fourth, we limited our dependent variable to fitness. Therefore, we treat different causes for fitness differences similar. However, it might be better to include deviation information to find root causes of these fitness differences.

A limitation of our evaluation is that no to-be models were available for the BPI logs, meaning that our evaluation results have to be interpreted carefully. We tried to mitigate this limitation by applying ABCD in a case with to-be model. However, we acknowledge that the insights of ABCD heavily depend on the availability of these models. This could be addressed by data-driven approaches for deriving to-be models, reducing the necessary effort for the organizations.

Finally, ABCD only identifies that a certain attribute value or combination of attribute values is correlated with process conformance, but it does not explain how the conformance is influenced. As discussed in Sect. 3.3, the next step could be to incorporate a post-processing that investigates the alignments of the sub-logs generated in the leaf nodes and analyzes where and how a deviation occurs.

## References

1. van der Aalst, W.: Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining. In: AP-BPM 2013. Springer (2013)
2. van der Aalst, W., Guo, S., Gorissen, P.: Comparative process mining in education: An approach based on process cubes. In: SIMPDA 2015. Springer (2013)
3. Adams, J.N., van Zelst, S., Quack, L., Hausmann, K., van der Aalst, W., Rose, T.: A framework for explainable concept drift detection in process mining. In: BPM 2021. Springer (2021)
4. Bolt, A., de Leoni, M., van der Aalst, W., Gorissen, P.: Business process reporting using process mining, analytic workflows and process cubes: A case study in education. In: SIMPDA 2015. pp. 28–53. Springer (2017)
5. Borrego, D., Barba, I.: Conformance checking and diagnosis for declarative business process models in data-aware scenarios. Expert Systems with Applications **41**, 5340–5352 (2014)
6. Burattin, A., Maggi, F., Sperduti, A.: Conformance checking based on multi-perspective declarative process models. Expert Systems with Applications **65**, 194–211 (2016)
7. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018)
8. Cheng, C.L., Shalabh, Garg, G.: Coefficient of determination for multiple measurement error models. Journal of Multivariate Analysis **126**, 137–152 (2014)
9. Chicco, D., Warrens, M., Jurman, G.: The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. PeerJ. Computer science (2021)

10. De Leoni, M., van der Aalst, W., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Information Systems **56**, 235–257 (2016)
11. Delias, P., Grigori, D., Mouhoub, M.L., Tsoukias, A.: Discovering characteristics that affect process control flow. In: EWG-DSS 2014. pp. 51–63. Springer (2015)
12. Delias, P., Lagopoulos, A., Tsoumakas, G., Grigori, D.: Using multi-target feature evaluation to discover factors that affect business process behavior. Computers in Industry **99**, 253–261 (2018)
13. Di Francescomarino, C., Ghidini, C.: Predictive Process Monitoring, pp. 320–346. Springer, Cham (2022)
14. Fani Sani, M., van der Aalst, W., Bolt Irondo, A., García-Algarra, J.: Subgroup discovery in process mining. In: BIS 2017. pp. 237–252. Springer (2017)
15. FAU, Chair of Digital Industrial Service Systems: Process Mining Software Comparison (2020), `https://www.processmining-software.com/tools/`
16. Ferreira, D., Vasilyev, E.: Using logical decision trees to discover the cause of process delays from event logs. Computers in Industry **70**, 194–207 (2015)
17. Gupta, M., Sureka, A.: Process cube for software defect resolution. In: APSEC 2014. pp. 239–246. IEEE (2014)
18. Hompes, B., Maaradji, A., La Rosa, M., Dumas, M., Buijs, J.C., van der Aalst, W.: Discovering causal factors explaining business process performance variation. In: CAiSE 2017. Springer (2017)
19. Horita, H., Hirayama, H., Tahara, Y., Ohsuga, A.: Towards goal-oriented conformance checking. In: SEKE 2015. pp. 722–724 (2015)
20. Mannhardt, F., de Leoni, M., Reijers, H., van der Aalst, W.: Balanced multi-perspective checking of process conformance. Computing **98**, 407—-437 (2016)
21. Mozafari Mehr, A., Medeiros de Carvalho, R., van Dongen, B.: Detecting privacy, data and control-flow deviations in business processes. In: CAiSE 2021. Springer (2021)
22. Munoz-Gama, J.: Conformance Checking and Diagnosis in Process Mining: Comparing Observed and Modeled Processes. Springer (2016)
23. Qafari, M., van der Aalst, W.: Case level counterfactual reasoning in process mining. In: CAiSE 2021. pp. 55–63. Springer (2021)
24. Rozinat, A., van der Aalst, W.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**, 64–95 (2008)
25. Saunders, L., Russell, R., Crabb, D.: The Coefficient of Determination: What Determines a Useful $R^2$ Statistic? Investigative Ophthalmology & Visual Science **53**, 6830–6832 (2012)
26. Scheid, M., Rehse, J.R., Houy, C., Fettke, P.: Data set for mobis challenge 2019 (2018)
27. Swinnen, J., Depaire, B., Jans, M.J., Vanhoof, K.: A process deviation analysis – a case study. In: BPM 2011. pp. 87–98. Springer (2012)
28. Tsoury, A., Soffer, P., Reinhartz-Berger, I.: How well did it recover? impact-aware conformance checking. Computing **103**, 3 – 27 (2021)
29. van Dongen, B.: BPI challenge 2017. `https://data.4tu.nl/articles/dataset/BPI_Challenge_2017/12696884` (2017)
30. van Dongen, B.: BPI challenge 2020. `https://data.4tu.nl/collections/_/5065541/1` (2020)
31. Zhang, Z.: Missing data imputation: Focusing on single imputation. Annals of translational medicine **4** (2016)