Impact-Aware Conformance Checking

Arava Tsoury(✉), Pnina Soffer(✉), and Iris Reinhartz-Berger(✉)

University of Haifa, Mount Carmel, 3498838 Haifa, Israel
{atsoury, spnina, iris}@is.haifa.ac.il

Abstract. Alignment-based conformance checking techniques detect and quantify deviations of process execution from expected behavior as depicted in process models. However, often when deviations occur, additional actions are needed to remedy and restore the process state. These would seem as further reducing conformance according to existing measures. This paper proposes a conformance checking approach which considers the response to unexpected deviations during process execution, by analyzing the data updates involved and their impact on the expected behavior. We evaluated our approach in an experimental study, whose results show that our approach better captures adapted behavior in response to deviations, as compared to standard fitness measurement.

Keywords: Conformance checking · Alignment · Data impact analysis · Business processes

1 Introduction

Business process models depict the expected and normative course by which processes are expected to be executed. Over the years, process mining techniques [16] were developed for gaining insights into business process behavior from event logs, which capture events that typically correspond to activities performed using an information system. Conformance checking is an area within process mining [6] that analyzes the relations and differences between the expected behavior, specified in a process model, and the actual behavior, reflected in an event log. Most conformance checking techniques (e.g., [1, 15]) focus on control flow aspects, addressing the ordering and flow of activities in the process. Other aspects have been also considered, e.g., [7, 9, 11] and [12] for checking conformance with respect to resource and data-related rules.

Conformance checking is done by comparison of observed behavior with the modeled process. Nevertheless, process models rarely capture the full range of possible behaviors. In particular, exceptions that may occur and possible compensation activities that may be needed due to errors are typically not described in the process model [4]. Hence, comparison may yield deviation from the prescribed process model, as well as involvement of additional data operations that may influence the process state. These may have further consequences in other parts of the process. In other words, when analyzing the event logs using existing conformance checking techniques, process executions can appear to be non-conformant, whereas in fact, they exhibit the expected behavior given the unexpected situation (e.g., of an error and its correction). In contrast, process executions, in which unexpected changes were not fully and appropriately

C. Di Francescomarino et al. (Eds.): BPM 2019 Workshops, LNBIP 362, pp. 147–159, 2019.
https://doi.org/10.1007/978-3-030-37453-2_13

handled, may appear to have better conformance, since the differences between the process model and the event log may be smaller. We claim that the consequences of such a change can be identified by analysing the impact of the changed data values on the process. In this paper, we introduce the concept of *impact-aware conformance checking*, which takes into consideration unexpected events and data changes during process execution for calculating a conformance score. The approach is based on the existence of two main sources: (1) the event log, which may include only basic process control-related information or additional data attributes; (2) the database transaction (redo) log that captures historical data operations performed on the database as a result of business process activities. We have already proposed combining these two sources in [13]. The approach suggested here relies on combining information from event and transaction logs and employing a data-impact analysis technique [14], which propagates an unexpected change in data values along the process model and returns a set of affected process elements. We further introduce a new measure and a new technique for impact-aware conformance checking.

The remainder of the paper is organized as follows. Section 2 presents a running example to illustrate and motivate the need for our approach. Section 3 is devoted to preliminaries required and premises of our approach. Section 4 presents the approach, whose evaluation is reported in Sect. 5. Section 6 discusses related work and, finally, in Sect. 7 we conclude and discuss limitations, raising future research directions.

2 Running Example

To motivate our approach, consider a sales process, in which customers order products that are shipped after payment. Figure 1 depicts a Petri net model of this process. Typically, the data used in the process is stored in a relational database, which includes tables related to customers, orders, shipping companies, employees, and so on.

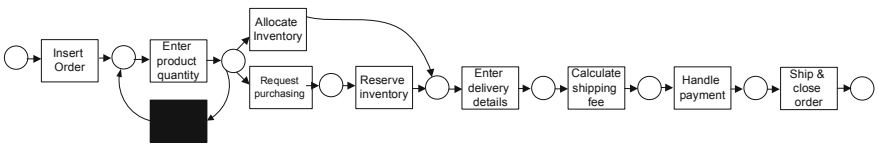


Fig. 1. The sales process model

Let us assume the customer asked to change the quantity of an ordered product *after* payment. This would require repeating the activity *enter product quantity* after *handle payment*. Now consider two possible traces (activities are marked by abbreviations):

(a) *<IO, EPQ, AI, EDD, CSF, HP, EPQ, AI, CSF, HP, SCO>*
(b) *<IO, EPQ, AI, EDD, CSF, HP, EPQ, AI, SCO>*

Clearly, both traces exhibit non-conformant behaviors, addressing unexpected changes after payment. They differ in that trace (a) includes an additional calculation of

the shipping fee and additional handling of payment, while trace (b) continues the process normally after performing the change. These differences imply that existing conformance checking methods will find that trace (b) is more conformant than trace (a), since the deviation in (b) is only in the execution of two activities (*enter product quantity* and *allocate inventory*), as opposed to four activities in trace (a) (*enter product quantity, allocate inventory, calculate shipping fee, handle payment*). However, after the unexpected change occurred, trace (a) is more appropriate in business terms than trace (b), since it handles potential consequences of the change in product quantity. Such a change may necessitate additional changes in the shipment arrangements, which may, in turn, lead to changes in the shipping fee. Thus, to properly handle such a change, calculating the shipping fee should be revisited, and, as a result, an additional payment may need to be handled before the process can complete.

To tackle the problems illustrated above, we introduce the notion of impact-aware conformance, for analyzing the differences between the expected behavior and the actual one, in scenarios where unexpected changes may occur. We first elaborate preliminaries and premises and then present the approach.

3 Preliminaries and Premises

As noted, event logs, which are commonly produced by process-aware information systems, are essential for process mining techniques [16]. The content of event logs varies from "thin" logs, which contain only an event label (representing an activity) and a case identifier, to "fat" logs, which may contain data items relevant to the events [10]. Below, we provide some definitions of the basic concepts.

Definition 1 - Event (e), trace (σ): An *event* e is a tuple of the form e = (*caseID, activity, timestamp,* d_1, ... d_n), where *caseID* uniquely identifies a single process execution; *activity* is the event label; *timestamp* holds the time in which the event occurs; $d_1, \ldots d_n$ are additional (optional) data items. A *trace* σ is a sequence of events referring to the same process execution, $\sigma = <e_1, \ldots, e_m>$.

Besides event logs, conformance checking requires process models against which the event logs are checked. A process model describes the sequences of activities to be performed for reaching a certain business goal [4]. Different representations of process models have been suggested [3, 12]. In this paper we use a Petri net representation.

Definition 2 - Process model (M): A process model (M) is a triplet $M = (P, T, F)$, where P is a set of places; T is a set of transitions; $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation connecting places and transitions.

The state-of-the-art conformance checking approaches construct an alignment between a process model M and a trace σ [2]. An alignment is represented as a two-row matrix, where the first row consists of trace events and the second row includes process model activities.

Definition 3 - Alignment (γ): An alignment γ of a process model $M = (P, T, F)$ and a trace σ is a sequence of pairs of the form (e_i, a_i) where $e_i \in \{e \mid e \text{ is contained in } \sigma\} \cup \{>>\}$, $a_i \in T \cup \{>>\}$, and if $(e_i \neq >>$ and $a_i \neq >>)$ then e_i. activity $= a_i$.

Three cases of alignment moves can be defined [6] as follows: (1) *Synchronous move*: involving a matching event and model activity $e_i \neq >>$ and $a_i \neq >>$. (2) *Model move*: a model activity is not matched by an event $e_i = >>$ and $a_i \neq >>$, and (3) *Log move*: an event is not matched by a model activity $e_i \neq >>$ and $a_i = >>$. The last two are generally termed asynchronous moves.

Different alignments may be possible for the same pair of trace and process model. Alignment-based conformance checking commonly constructs "optimal" alignments, with respect to a given cost function that assigns costs to asynchronous moves. The optimal alignment minimizes the total cost of moves. Based on the optimal alignment, conformance can be quantified using the notion of fitness [15].

Definition 4 - Fitness: Given an optimal alignment γ, whose cost is $K(\gamma)$, the *fitness* is calculated as: $Fitness(\gamma) = 1 - \frac{K(\gamma)}{K(\gamma_R)}$, where $K(\gamma_R)$ is the maximal cost of a reference alignment γ_R, which is computed by concatenating moves in log for all events of σ with the alignment of the empty trace.

The basic alignment techniques and fitness measurement relate to a control-flow perspective, correlating a trace with a process model. Recent approaches suggest a multi-perspective data-aware alignment, considering also data and resources in the event log, compared to a model which includes these elements (e.g., [9]). Respective cost functions penalize deviations in data operations as well as in activity executions. Accordingly, optimal alignments are obtained and used in a fitness measure. Data-aware conformance checking is hence more powerful in detecting non-conformant behavior and explaining it. However, its applicability is restricted by the inclusion of data items in the log. Typically, when an event log is created, only a subset of data items used and manipulated by the process is included. In realistic settings, the total number of data items is too large to be included in an event log that should serve for mining and analysis. The selection may reflect expectations of relevance for analysis purposes, but the result is necessarily partial.

To overcome this incompleteness, we suggest using a control-flow perspective as a baseline for alignment, complemented by a *transaction log* for additional explanation and fine-tuning of detected deviations. Transaction logs are typically produced by database management systems and can be provided in diverse formats, depending on the platform. Yet, their structure can be generalized as follows.

Definition 5 - Transaction log: A Transaction log is a set of tuples of the form (*transactionID, beginTime, endTime, operation, caseObject, caseID, attribute, newValue*), where *transactionID* uniquely identifies the transaction; *beginTime* and *endTime* are the timestamps of the transaction's beginning and end, respectively; *operation* $\in$ {insert, update, delete} specifies the change type; *caseObject* and *caseID* identify the changed data item; *attribute* and *newValue* specify the change essence.

The relation between a transaction log and a corresponding event log can be established through two common attributes: (1) the case identifier – which explicitly appears in both event and transaction logs; and (2) the timestamp – the event timestamp should be within the transaction's time frame, assuming that writing to the database is not delayed. Proposals have been made (e.g., [10, 13]) to use a combination of an event log and a transaction log, based on mapping operations between these two sources.

In this paper, the use of a transaction log is to retrieve the data items that have been changed by a specific activity (event). Our study assumes that database operations and activities execution are both recorded. Also, it assumes that data operations are always executed in the context of process activities.

A last element needed for our approach is a mechanism for analyzing data impacts on the process. Data impact analysis addresses dependencies among process elements, stemming from the required data flow. Considering a data item whose value may be changed by a deviating activity, the activities included in the partial trace before the deviation, which were affected by the data item, are its *impact* in the partial trace. The approach suggested in [14], for example, analyzes the effects of a single data item (an attribute or an object) on other process elements, including activities.

Definition 6 - Data Impact (DI): Given a data item[1] d and a trace σ, the set of activities represented by events in σ and affected (either directly or indirectly) by the value of d are termed the *data impact* of d in σ, and marked DI(d, σ).

Returning to our running example, consider the *ordered quantity*, which is determined in the activity *enter product quantity*, and assume its value has been updated unexpectedly after a partial trace σ = <*IO, EPQ, AI, EDD*>. Changing the quantity at this phase may require changes in the inventory allocation and possibly also in the delivery details (a different truck may be needed). Accordingly:

DI(*Ordered quantity, σ) = {AI, EDD}*.

4 The Approach

The main idea of our suggested approach is that once an unexpected change occurs, manifested as a log move, the expected behavior may no longer be the behavior prescribed by the process model. Rather, additional actions may be required as compensation, and these should be augmented as part of the expected behavior, forming a new basis for conformance checking. We now discuss how the expected behavior is recalculated following a deviation from the prescribed behavior.

4.1 Expected Behavior After Initial Deviation

Recall that a *log move* represents an activity that was executed but could not be aligned with the process model, namely, a deviation from the expected behavior. This deviation might occur for many reasons, such as non-compliance of the user, or due to exceptional and unexpected situations that necessitate an ad-hoc change. To understand the essence of the activity classified as a log move, we seek the data operations involved. For an in-depth analysis of the impact of deviating activities, we turn to the full set of data operations (e.g., insert, update, delete) that took place, and are typically available in the transaction log. Consider that a deviating activity a updated the values of a set of data items, denoted by *Aff(a)*. This means that each data item in *Aff(a)* was updated by the event representing a (in the same period of time and by the same case identifier).

[1] A data item is an attribute or an object, whose impact is of interest.

Now assume that at least some of these data items have been used earlier in the trace, making impacts on activities and on path selection decisions. This may require revisiting and correcting parts of the process activities. The activities that may potentially be revisited as a response to the deviation are identified by analyzing the data impact of the data items in *Aff(a)*, considering the trace preceding the deviation in a.

Definition 7 - Response set (RS): Given a trace σ, followed by an activity a classified as a log move in an alignment γ, the ***response set*** to the log move a is $RS(a, \sigma) = \bigcup_{d \in Aff(a)} DI(d, \sigma)$.

Turning to the expected behavior of the process following a deviation (log move), it should include activities from the response set in addition to the ones that correspond to the process model. We cannot, however, anticipate the order in which responses will be attended to. Rather, if the remaining trace includes activities which are not aligned with the process model (additional log moves) but are in the response set, we interpret them as responses and consider them as part of the expected behavior of the process. In other words, assume a baseline expected behavior as all compliant process traces that include the partial trace σ. The expected behavior following an initial deviation a will extend the baseline and also include the set of response activities *RS(a, σ)*. Listing 1 provides the algorithm for retrieving the response set, correlating the event log with a transaction log which holds all the data operations performed.

Algorithm 1: Retrieve Response Set (RetrieveRS)

Input: σ is a partial trace, *a* is a log move occurring immediately after σ

Output: RS is the set of response activities

```
1. AFF = null
› retrieve related data items in the transaction log (based on case
id and timestamp)
2. AFF = retrieveDataItems(a.caseId, a.timestamp)
3. if AFF is empty then
4.    return › no data items have changed
5. end
6. RS = ∅
›Iterate over all data items in AFF to retrieve their impact
7. for each d∈ AFF do
8.      RS ← RS ∪ DI(d, σ)
9. end
10. return RS
```

Listing 1. Algorithm for retrieving response activities for a specific log move

4.2 Impact-Aware Alignment and Fitness

With the extension of the expected behavior, we now turn to adapt the alignment and the fitness measure. Note that calculating a new optimal alignment is not possible since the expected behavior is now extended by an unordered set of response activities. We hence modify the initial given alignment to cater for the extended expected behavior. For this task, the result set and the given alignment are compared. Log moves in the

alignment, whose activities are included in the response set, are marked as *response moves* (denoted by ρ). These marks allow for extending the definition of alignment to be impact-aware.

Definition 8 - Impact-Aware Alignment: An *impact-aware alignment* of a process model M = (P, T, F) and a trace σ is an alignment, i.e., a sequence of pairs of the form (e_i, a_i), where a_i can be a response move (i.e., $a_i \in (T \cup \{>>, \rho\})$), and if $(a_i = \rho)$ then there is a preceding log move a_j =>> (j < i) such that e_i.activity ∈ RS(a_j, <$e_1 \ldots e_{j-1}$>).

While transforming the given alignment to an impact-aware one, we keep track of the activities in RS, but also remove them from RS when encountered along the alignment. Note that this removal takes place whether the activity is part of the original process model (namely, corresponds to a synchronous move in the alignment) or not. The rationale is that an activity is included in RS if its data operations may be required as a response to some deviation. When that activity is performed afterward, additionally to or in-line with the process model, the response is considered done, and the activity can be removed from RS. Having gone through the entire alignment, the activities remaining in RS are *missed response activities*.

To illustrate, consider the following alignment γ of trace (a) in the running example.

IO	*EPQ*	*AL*	*EDD*	*CSF*	*HP*	*EPQ*	*AL*	*CSF*	*HP*	*SCO*
IO	*EPQ*	*AL*	*EDD*	*CSF*	*HP*	>>	>>	>>	>>	*SCO*

For the first log move, which corresponds to *EPQ*, the calculated response set is *RS = {AL, CSF, HP}*. Hence, the impact-aware alignment corresponding to γ is:

IO	*EPQ*	*AL*	*EDD*	*CSF*	*HP*	*EPQ*	*AL*	*CSF*	*HP*	*SCO*
IO	*EPQ*	*AL*	*EDD*	*CSF*	*HP*	>>	ρ	ρ	ρ	*SCO*

In this case the RS remains empty after creating the impact-aware alignment. This means that all expected corrective activities have been performed. This would not be the case with trace (b) in our running example.

As a last note, deviations and unexpected situations may occur more than once when a process is executed. We repeat the analysis for every log move which is not interpreted as a response to a previous one in a trace, recalculating the expected behavior of the process iteratively. Listing 2 provides the algorithm which analyzes an alignment and transforms it to an impact-aware alignment. Note, Algorithm 2 uses Algorithm 1 (Retrieve Response Set – see Listing 1).

We now turn to the calculation of a fitness measure based on the impact-aware alignment. We term this measure *Impact-aware fitness*. For this, the following generic cost function can be used for calculating the cost of an alignment:

$K(\gamma) = C_M * |RS| + \sum CostOfMove$, Where:

$$CostOfMove = \begin{cases} 0 & Synchronous\ move \\ 1 & Log\ move\ or\ Model\ move \\ C_\rho & Response\ move \end{cases}$$

$C_\rho \in [0,1]$ and $C_M \in [0,1]$ are factors indicating the cost associated with a response move and with a "missed" response that remains in RS, respectively; |RS| is the number of elements that remain in the response set after making the alignment impact-aware. With this cost function, the Impact-aware fitness of γ is calculated using the standard fitness definition (see Definition 4). In our running example, if we consider the cost of a response move C_ρ as 0 and the cost of a missed response C_M as 1, the impact-aware fitness of trace (a), where response to a deviation was handled, is 0.94 while its standard fitness is 0.76. The impact-aware fitness of trace (b) (response to deviation not handled) is 0.8 while its standard fitness is 0.87.

Algorithm 2: Construct impact-aware alignment

Input: γ is an alignment

Output: γ' is an impact-aware alignment

```
1. RS = ∅; γ'=γ        > Initialize variable
› Iterate over γ, maintain RS and create γ'
2. for each (e_i. a_i) ∈ γ, i=1…n, do
3.  Response=false     > Initialize a Boolean variable Response
4.    if (e_i.activity ∈ RS) then
5.         RS←RS  - {e_i.activity}
6.        Response=true
7.    end if
8.    if a_i = >> then
9.          if Response then γ' . a_i = ρ
10.         else RS ← RS∪RetrieveRS(<e_1…e_i-1>, e_i.activity)
11.   end if
12. end for
13. Return γ'
```

Listing 2. An algorithm for transforming an alignment to an impact-aware alignment

5 Evaluation

For feasibility evaluation, we conducted an experimental study using simulated data[2]. We compared our results to those of a standard conformance checking technique [2]. Below, we elaborate on the evaluation design & execution, as well as on the results.

5.1 Evaluation Design

We used the relatively simple sales business process, depicted in Fig. 1. We further defined several scenarios, some conformant and some engaging deviations. When deviations were engaged, we controlled for required compensations (with/without), and for the extent to which these compensations were handled. Last, we generated scenarios where random noise was added. The numbers of simulated cases for these types are

[2] The simulated data is available at https://drive.google.com/drive/folders/1RnPxzjO1chO8NEtA3tCCcnKdOuH6_Uhf?usp=sharing

listed in Table 1. For each scenario type, we analyzed the expected differences between standard and impact-aware fitness (column 5 in the table, where standard fitness is marked as F, and impact-aware fitness – as IaF). Particularly, we expect that the impact-aware fitness in *fully handled* scenarios to be higher than the standard fitness, since more compensation activities may imply a greater discrepancy between the model and the event log, thus a lower (standard) fitness. Impact-aware fitness, on the other hand, will count those activities as response moves, whose cost is lower than that of log moves. Following a similar line of thinking, the impact-aware fitness in *not handled* scenarios will be lower than the standard fitness due to missing response activities. For *partially handled* scenarios, the impact-aware fitness depends on the number of compensation activities that have been performed vs. the number of missing responses. For *no required handling* scenarios, we expect both types of fitness to be equal, whereas for *conformant* scenarios, we expect them to be equal to 1. Finally, for *random noise* scenarios, in which activities are performed in a non-conformant manner and without reason, we cannot predict the differences between fitness values.

Following these types of scenarios, we simulated an event log of 70,805 events and a related transaction log[3] with 121,139 records. Overall, we simulated more than 7700 cases, whose distribution among the scenario types is also presented in Table 1.

Table 1. The simulated scenario types

Required compensation	Scenario type	Example in the sales process	Expected fitness differences	# cases
With	Fully handled	• Increase ordered quantity after payment • Decrease ordered quantity after payment • Change shipping type after payment	F < IaF	1141
	Partially handled		Unpredictable	826
	Not handled		F > IaF	903
Without	No required handling	• Update shipping address after payment • Update payment type after payment	F = IaF	627
	Conformant		F = IaF	4173
	Random noise	• Activities are added/skipped randomly	Unpredictable	40

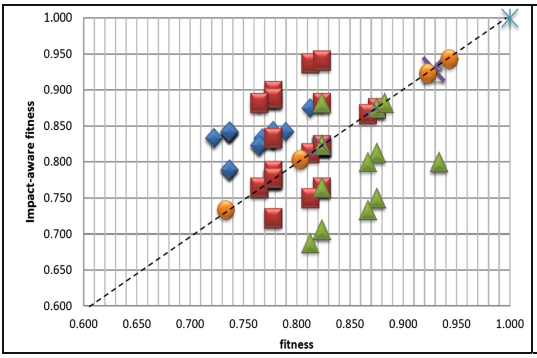
5.2 Results

For simplicity we considered the cost of a response move in the impact-aware fitness as 0 and the cost of a missed response as 1.

Figure 2 presents a comparison of the standard and impact-aware fitness values for the different scenario types (shown as different series). Points on the diagonal line are

[3] The transaction log was simulated as an audit table, tracking all changes done to the database.

cases where the two types of fitness are equal, points above are where impact-aware fitness is higher than the standard fitness, and points below are where the opposite holds. As can be seen, impact-aware fitness is higher than standard fitness in situations where the deviations were fully or partially handled. In situations where the deviations were not handled at all, the standard fitness is higher than the impact-aware fitness.



Scenario	Avg F	Avg IaF
Fully handled	0.77	0.84
Partially handled	0.8	0.83
Not handled	0.85	0.81
No required handling	0.87	0.87
Conformant	1	1
Random Noise	0.9	0.9

Fig. 2. A comparison of the standard and impact aware fitness for the different scenarios

In summary, the obtained results confirm our expectations of how the different scenario types would be assessed in terms of impact-aware fitness. They indicate that impact-aware fitness better captures the extent to which consequences of deviations are handled, as compared to standard fitness.

6 Related Work

Many conformance checking techniques have been proposed over the years, and they mostly focused on control flow aspects. More recently, additional perspectives have been addressed, including data, resources, and time (e.g., [7, 9, 11, 12]). In particular, optimal alignments of a log and a model have been computed using cost functions that consider deviations on these perspectives as well as the control-flow ones. For performing analyses of this kind, an event log with the relevant data attributes is required (i.e., the role of the activity executer or specific values of data attributes).

Our main interest is on studies which take a step toward an explanation of non-conformance. In [8] the authors propose a conformance checking approach which uses declarative models as a basis. Such models specify constraints over process execution and allow flexibility in the expected behavior. Note that the constraints relate to the control flow of the process. They may possibly reflect data-related dependencies among activities, but this is not apparent and thus not considered explicitly. Attempting to provide explanations of non-conformance, the work in [3] focuses on an automated inference of a cost function from an event log as a basis for obtaining an optimal alignment. The data perspective is captured as a process state. Compared to our approach, the main limitation is in the need to rely on the data available in the event log. This limitation does not exist in [4], where data logs (a version of transaction log)

and event logs connect the data and process perspectives. The authors propose an algorithm to construct inter-level alignment by linking the data operations recorded in the database log with activities in the control flow alignment. Furthermore, they use a CRUD (create, read, update, delete) matrix, depicting the data operations that can be performed by each activity in the process. As opposed to our approach, they do not consider the impact of deviations on the expected behavior and thus responses to deviations are not addressed. Another recent study which concerns explainability is [5]. In contrast to the common focus of conformance checking approaches (including ours), the focus of that work is on the explainability of conformant traces. The authors suggest metrics that can distinguish between conformant traces and indicate their differences.

To sum up, the literature in the field of conformance checking is rich and diverse. However, the impact of deviations and their data operations on the expected behavior has not been explicitly addressed so far, as is done in this paper. Such an analysis is important when unexpected deviations during process execution occur.

7 Conclusion

This paper has presented a conformance checking approach which considers possible changes in the expected behavior in response to unexpected changes and deviations during process execution. Besides the technical solution we propose, the paper makes two main contributions. First, it highlights the response and compensations that may need to follow unexpected situations, which are manifested as a deviation or a log move. Existing conformance checking techniques do not recognize this need and thus compensation actions, which are normative in business terms, are considered as non-conformant and result in poor conformance assessments (e.g., fitness values).

Second, it proposes a novel use of a combination of an event log and a transaction log. Rather than relying on availability of data in the event log, our approach takes a control flow-based alignment as a baseline, and only upon detection of a deviation, it seeks its data reflection in the full transaction log. By this, two main advantages are achieved: (1) overcoming the need to rely on a partial set of preselected data items that may be available in the event log; (2) avoiding the complexity of dealing with the full set of data throughout the alignment.

The suggested approach uses an impact analysis mechanism. Currently, such mechanisms exist as a stand-alone system, whose availability may form a limitation for the approach. Nevertheless, this can be further integrated into process mining environments and serve for additional analyses. Another limitation is that the approach is based on a given alignment; different alignment algorithms and cost functions may yield different alignments, which may also affect our results. This limitation, however, is inherent in the nature of alignments, which can be optimized but are not considered absolute.

We note that the evaluation reported here used synthetic data, where both the event log and the transaction log were simulated. Additional and extended evaluation of our approach is still needed, using real-life event and transaction logs. In future work, we seek to extend the approach to consider the data-flow along the activities, using a transaction log that stores data values and their changes.

Acknowledgment. This research is supported by the Israel Science Foundation under grant agreements 856/13 and 669/17.

References

1. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.: Conformance checking using cost-based fitness analysis. In: 2011 IEEE 15th International Enterprise Distributed Object Computing Conference, pp. 55–64. IEEE (2011)
2. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.: Memory-efficient alignment of observed and modeled behavior. BPM Center Report, 03-03 (2013)
3. Alizadeh, M., De Leoni, M., Zannone, N.: Constructing probable explanations of nonconformity: a data-aware and history-based approach. In: 2015 IEEE Symposium Series on Computational Intelligence, pp. 1358–1365. IEEE (2015)
4. 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)
5. Burattin, A., Guizzardi, G., Maggi, F.M., Montali, M.: Fifty shades of green: how informative is a compliant process trace? In: Proceedings of CAiSE (2019)
6. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking: Relating Processes and Models. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99414-7
7. de Leoni, M., Van Der Aalst, W.M.: 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, vol. 8094, pp. 113–129. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40176-3_10
8. de Leoni, M., Maggi, F.M., van der Aalst, W.M.: An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data. Inf. Syst. **47**, 258–277 (2015)
9. de Leoni, M., Van Der Aalst, W.M., Van Dongen, B.F.: Data-and resource-aware conformance checking of business processes. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds.) Business Information Systems. LNBIP, vol. 117, pp. 48–59. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30359-3_5
10. de Murillas, E.G.L., Reijers, H.A., van der Aalst, W.M.: Connecting databases with process mining: a meta model and toolset. Softw. Syst. Model. **18**, 1–39 (2018)
11. Ramezani Taghiabadi, E., Fahland, D., van Dongen, B.F., van der Aalst, W.M.P.: Diagnostic information for compliance checking of temporal compliance requirements. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 304–320. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38709-8_20
12. Taghiabadi, E.R., Gromov, V., Fahland, D., van der Aalst, W.M.P.: Compliance checking of data-aware and resource-aware compliance requirements. In: Meersman, R., et al. (eds.) OTM 2014. LNCS, vol. 8841, pp. 237–257. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45563-0_14
13. Tsoury, A., Soffer, P., Reinhartz-Berger, I.: A conceptual framework for supporting deep exploration of business process behavior. In: Trujillo, J., et al. (eds.) Conceptual Modeling. ER 2018. LNCS, vol. 11157, pp. 58–71. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00847-5_6
14. Tsoury, A., Soffer, P., Reinhartz-Berger, I.: Towards impact analysis of data in business processes. In: Schmidt, R., Guédria, W., Bider, I., Guerreiro, S. (eds.) BPMDS 2016. LNBIP, vol. 248, pp. 125–140. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39429-9_9

15. Van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **2**(2), 182–192 (2012)
16. Van Der Aalst, W.: Process Mining: Discovery, Conformance and Enhancement of Business Processes, vol. 2. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19345-3