The published version is available at https://doi.org/10.1109/EDOC49727.2020.00031

# Data-driven Improvement of Online Conformance Checking

1st Florian Stertz
*University of Vienna*
*Faculty of Computer Science*
Vienna, Austria
florian.stertz@univie.ac.at

2nd Juergen Mangler
*University of Vienna*
*Faculty of Computer Science*
Vienna, Austria
juergen.mangler@univie.ac.at

3rd Stefanie Rinderle-Ma
*University of Vienna*
*Faculty of Computer Science*
Vienna, Austria
stefanie.rinderle-ma@univie.ac.at

***Abstract*—Conformance checking takes a process model and a process log as input and quantifies the degree of conformance between both. This allows a comparison between the intended behavior represented by the model and the actual behavior captured by the log and is useful for many applications such as auditing. Existing approaches calculate conformance as follows: each deviation between model and log is corrected by an alignment, e.g., inserting a missing event to the log, that has a standard per-deviation cost of 1. While deviations in the model can be handled this way, there is no way to differentiate between intended (e.g., ad-hoc repair of instances) and unintended (e.g., security breaches) deviations. Hence this work proposes an advanced cost function, that allows for per-deviation adjustments of the per-deviation costs. By inspecting how the data elements of subsequent tasks are affected, it becomes possible to automatically increase or decrease the per-deviation costs of 1, thus allowing for an automatic classification of deviation causes. The proposed approach works offline and online (i.e., at runtime) and is evaluated based on a real-world dataset from the manufacturing domain.**

***Index Terms*—Process mining and business analytics, Online Conformance Checking, Logging Errors, Data-driven Alignment Costs**

## I. Introduction

In the field of process mining, *conformance checking* (CC) is designed to determine if the logged process execution of a business process correctly represents the desired execution specified by a process model [8]. For this, CC takes a process model and a process log as input and calculates the conformance between model and log. A conformance of 1 means that the behavior described by the model is perfectly reflected in the log and vice versa. A conformance $< 1$ indicates deviations in the behavior described by the model and reflected by the log. Then a conformance deviation (CD) has occurred. It is crucial to detect such CD and to investigate their reasons in detail in order to, for example, be able to distinguish intended deviations and undesired ones. The latter might hint to security breaches or ad-hoc changes in process instances due to problems during process execution.

Different types of CC can affect different artefacts, as depicted in Fig. 1. More precisely, CC can detect deviations that occur due to structural and semantic aspects. While structural CC deals with missing/additional events or data, semantic CC deals with deviations in timing, data or resources (i.e. irregularities regarding machines, humans or computing resources that are used while enacting process activities).

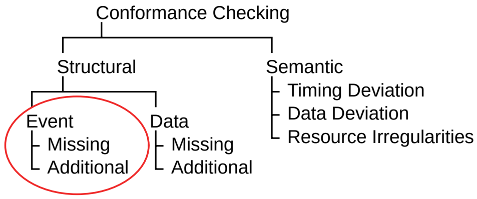


Fig. 1. Conformance Checking Types and Affected Artefacts

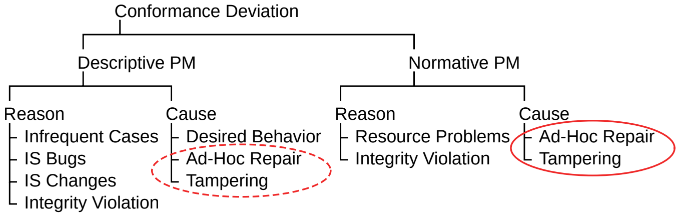


Fig. 2. Scenarios for Conformance Deviations (PM: Process Model)

Figure 2 shows resons and causes which might lead to CD. An automatic differentiation between those causes as well as the quantification of individual deviations is the main motivation for this paper.

Process models can be descriptive, i.e., described by a rule set consisting of rules defining the order of tasks for example, allowing execution of more infrequent cases, since the rule set typically is not covering every option possible. On the other hand, process models can be normative, where the order of the events is strict and each execution sequence not following the process model is not correct.

In a scenario with a **descriptive process model**, Information Systems (IS) with hard coded process logic contribute to an event log. These IS may contain logic for infrequent cases, that are not yet fully described by the process model. These IS may furthermore contain bugs (erroneous code), or they may be still under development introducing new behaviors. Lastly the underlying IS may be compromised, leading to security violations, or tampering. With CC it is possible to point out deviations which may then be attributed to one of the aforementioned reasons.

A scenario with a **normative process model** on the other hand is much less fuzzy. It can be assumed that the process model is actively enacted by some kind of process engine, thus most of the time the log will always be perfectly aligned with the PM. It can be furthermore assumed that PM changes only occur in line with CC changes. Thus the causes for CDs can typically be attributed to either problems with resources or security violations (tampering).

While previous publications (e.g. [20]) concentrated on semantic aspects, this paper will concentrate strictly on missing/additional events (see ○ in Fig. 1). The following research questions have been defined:

**RQ1:** *How can conformance deviations be quantified and classified based on data elements?*

**RQ2:** *What observable properties exist that can be used to adjust how severe the presence/absence of an event is?*

**RQ3:** *Can the quantification of a conformance deviation be used for automatic classification of possible causes?*

This paper proposes **a novel approach to quantify the overall conformance deviation cost of a single process execution** for a given PM and a given log, based on per-deviation cost values assigned to missing / additional events occurring in a process log. These per-deviation cost values can then be automatically adjusted based on causes shown in Fig. 2. While *ad-hoc repair* typically is conducted in a way that it does not affect any subsequent events, *tampering* can lead to observable effects in the log. Thus, based on a set of well-defined effects observable in the process log, the cost values may be either decreased on increased. The per-deviation costs contribute to advanced cost function which yields a CD cost value. The CD cost value can then be utilized for automatic classification of causes as depicted through ○ in Fig. 2. In addition, this approach can be applied on an event stream as well, instead of a process log, to detect tampering as soon as possible, therefore help the user repair the process more efficiently.

While the proposed approach is valid for both, descriptive and normative PMs, the evaluation has been conducted based on a normative PM and a process log produced by a process engine. For the evaluation an artifical dataset ([1]), as well as a real-world dataset from the manufacturing domain ([2]) have been used.

In Sec. II the required fundamentals on process mining and conformance checking are established. Section III introduces the advanced cost function and a possible implementation of a algorithm incorporating it. The contribution is then evaluated and discussed in Sec. IV. Related work in this field is presented in Sec. V while Sec. VI presents conclusions and future work.

## II. Fundamentals

In this section, selected fundamentals of process mining (cf. [3]) necessary for the subsequent conceptual contributions are presented, i.e., event logs, process models, conformance checking using alignments, and event streams. We illustrate the fundamentals based on an example from the medical domain.

Figure 3 shows a process model of typical tasks that have to be done for a patient in a hospital on a daily basis. At first, the patient is asked about his or her general state of health. Afterwards blood pressure and heart rate are measured in parallel. At the end drugs are administered according to the patient's treatment plan[3].

**Event Log:** An event log in process mining is typically defined in the XES format [1]. Every time a task is executed, it is logged, usually containing information on the patient as the trace id, the actor, the time, the event name, and other data elements, like the blood pressure or heart rate. A trace is a sequence of all events relating to the same trace id, so in our example all events for one patient on one day are a trace. An event log is a sequence of traces.

**Conformance Checking:** Conformance checking controls if a trace of an event log was correctly executed when compared to a given process model. Based on the process model depicted in Fig. 3, the following two traces can be produced: (Administer Drugs, Check Blood Pressure, Check Heart Rate, Check General State of Health) and (Administer Drugs, Check Heart Rate, Check Blood Pressure, Administer Drugs, Check General State of Health). Figure 4 shows two traces $t_1$ and $t_2$ of a given log. While $t_1$ is already perfectly aligned to one of the possible sequences of the model, $t_2$ is not as the two events Check Heart Rate and Administer Drugs) appear in the wrong order. Two possible alignment for $t_2$ are shown in Fig. 4.

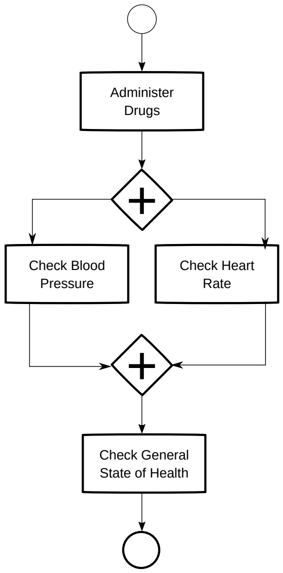


Fig. 3. Process Model for Patient Treatment in The Medical Domain

A row in Fig. 4 represents moves from one column to the next one. If the same event is present in both rows, it is called

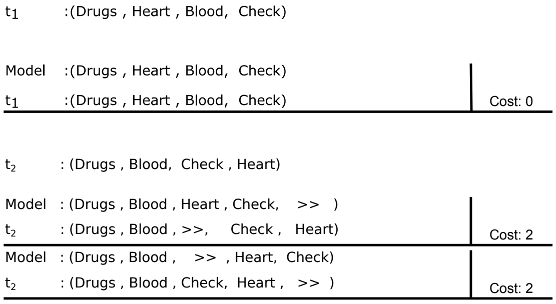
[1] http://gruppe.wst.univie.ac.at/data/edoc.xes

[2] http://gruppe.wst.univie.ac.at/data/timesequence.zip

[3] Process aware treatment plans are elaborated in [11]

a synchronous move. If there is an event in the aligned trace but a skip symbol ($\gg$) in the model, it is called a log move. On the other hand, if there is an event in the model and $\gg$ in the aligned trace, it is called a model move [2].

As can be seen in Fig. 4 with $t_2$, there is sometimes more than one possible alignment for one sequence. `Check General State of Health` and `Check Heart Rate` appear in the wrong order. This trace can be aligned by either using a model move, i.e., $\gg$ in the trace, followed by a log move, i.e., $\gg$ in the model or by using a log move at first, followed by a model move. Since there could be a plethora of possible alignments for a given trace and a process model, a distinction has to be made to find the best alignment. It could be viable to just count the number of asynchronous moves, but often times log and model moves have a different impact on the compliance of a trace. It may be more severe to execute an event twice than to not execute it at all. Therefore a standard cost function is introduced, where a cost for every move is assigned in the alignment [8]. In our approach we are using a simple cost function, that assigns the per-deviation cost of 0 for synchronous moves, 1 for non synchronous moves.



$t_1$ :(Drugs , Heart , Blood, Check)

Model :(Drugs , Heart , Blood, Check)
$t_1$ :(Drugs , Heart , Blood, Check) | Cost: 0

$t_2$ : (Drugs , Blood, Check , Heart)

Model : (Drugs , Blood , Heart , Check, >> )
$t_2$ : (Drugs , Blood , >>, Check , Heart) | Cost: 2

Model : (Drugs , Blood , >> , Heart, Check)
$t_2$ : (Drugs , Blood , Check, Heart , >> ) | Cost: 2

Fig. 4. Possible Alignments for Two Traces

`Event Stream`: An event log contains traces, which consist of a sequence of events. All the information on the execution of a specific trace is available since the execution is typically already finished (offline approach). Event streams, by contrast, create an event immediately, after an event has been executed. [22], [6] perform conformance checking on an event stream rather than on an event log (online approach). The main difference in working with an event stream, is that the execution of a trace is ongoing and events can still be appended to a sequence [21]. The approach presented in this paper can be applied in an online and offline manner.

This work exploits *process histories* [19] to discover process models. A process history $HP := < M_0, M_1, .., M_n, .. >$ contains a list of viable process models $M_i, i = 0, 1, ..$ that reflect the natural evolution of a business process $P$. In addition to the models, the process history enriches a process model with the expected data values for the data elements attached to an event [20], e.g., the expected blood pressure for a patient on a daily routine or even a sequence of measurements stored as a time sequence [21], where a time sequence is defined as a sequence of time-stamped data [10].

## III. Contribution

In this section the main contribution is explained in detail, i.e., the definition of the advanced cost function and how to gather the relevant data out of a process execution log.

### A. Advanced Cost Function

Conformance checking aims to align a sequence of events of a process log to a possible event sequence of a process model. For every deviation, i.e., an unexpected event in a sequence, a cost for the deviation is assigned. The alignment cost of a event sequence of a process log consists of the sum of all deviation costs. Currently, the cost for a deviation equals 1, whereas correctly fitting events have a cost of zero assigned [8].

While this cost function is generic, it does not use any information on data elements of events at all, an event simply is visible in a process execution log or not. We are introducing a new generic advanced cost function that aims to use all of the available information of the events in a process execution log and improve the results of conformance checking.

This approach is focusing on deviations, i.e., a missing or additional event in the process execution log to align to a possible sequence of a process model. To achieve this, acceptable values are gathered during the discovery of the process model. Acceptable values are, values for data elements that are compliant to the current process model. At the end of the section, the finding of acceptable values depending on the type of the data element is explained. When deviation is detected in an alignment, the data elements of the event after the deviation using a synchronous move are taken into consideration. If the values of the data elements result in correct values, even though an event is missing, the per-deviation cost of the preceding deviation should be reduced, since it indicates an error in logging if an event is missing in the log, or a successful repair of the process instance if an event is added to the log. Thus we define the following advanced cost function:

Since the advanced cost function is applied on a deviation using the data elements of the succeeding event, the standard cost function is used before and the per-deviation cost of the alignment is then altered using the advanced cost function. $b_m$ represents the base cost for a deviation, often 1. Since more than one data element can be attached to an event, the effect of specific data elements can be controlled. $D$ contains sets of all acceptable values for each data element attached to an event. The sets of acceptable values for each data element is calculated using a process execution log as a test set or while discovering the process model from an event stream, which is explained in detail at the end of this section. The function $\chi$ is the indicator function and checks if the value for a data element, is correct by checking if the value is in a set of acceptable values $D$. The indicator function returns 1 if the value is present and 0 if not. The result is then multiplied by a weight, $\omega$. This weight allows user to distribute the impact of data elements. The value of a weight is defined between $[0, 1]$ and the sum of all weights must be 1.

*Definition 1:*

$$advanced\_cost\_function(x) = \begin{cases} 0, & \text{if no deviation} \\ b_m - \sum_{n=0}^{len(data\_elements)} (\omega_n * b_m * \chi_{D_n}(data\_elements_n)), & \text{otherwise} \end{cases}$$

Let $b_m$ be the base cost per deviation and data_elements, a set of selected data_elements attached to an event. $D$ is a set of acceptable values for the data elements for this event, $\omega$ any number between 0 and 1, and $\chi$ the indicator function. This function returns the altered cost for a preceding deviation.

The base assumption here is the usage of a process execution engine, which does not allow unwanted events to be executed. If a deviation is still being detected, then there are 2 potential reasons for this deviation. Either, (a), an event is missing in the log. This could be the due to an error in the log or, worse, an attack on the process execution engine, skipping events. On the other hand, if (b), an additional event is found in the process execution log, it represents a repair event or an attack as well. If the data elements of the succeeding event contain acceptable values, an error in the log or a successful repair attempt is assumed and the conformance deviation cost of the alignment reduced. If the data elements do not contain acceptable values, an attack or unsuccessful repair attempt is assumed and the cost of the alignment is not reduced. Since more than one data element can be linked to an event, each of this data elements is inspected on its own.

### B. Implementation

```
Input: $M$: Process Model with information on
         acceptable values for data elements
   $T$: A Trace, containing events of a process execution
   $\kappa$: maximum of consecutive deviations, $\geq 1$
Result: $C$: Cost for the alignment of $T$ with $M$.
1  $alignment$ = conformance_checking(T)
2  $deviations$ = [ ]
3  for $move$ in $alignment$ do
4      // iterate over every move in the alignment
5      if move.type == Deviation then
6          $deviations$.append(move)
7          if len(deviations) ¿ $\kappa$ then
8              $deviations$.shift
9      if move.type != Deviation then
10         $advanced\_cost$ =
            advanced_cost_function($move.event$) // the
            event of the move
11         for $m$ in $deviations$ do
12             m.cost = $advanced\_cost$
13         $deviations$ = [ ]
14 $C$ = 0
15 for $m$ in $alignment$ do
16     $C$ += $m$.cost
17 return $C$
```

**Algorithm 1:** Finding Cost of Alignment

Algorithm 1, shows a possible implementation of the elaborated approach. As parameters a process model, a trace and $\kappa$ are needed. The parameter $\kappa$ is needed to define, how many none synchronous moves in a row can be affected by the advanced cost function, since depending on the process and the process domain it may not seem feasible to reduce the per-deviation costs for lots of deviations caused by data elements of the event of the next synchronous move. At first an alignment is calculated using a standard cost function. Afterwards a loop iterates through every move of the alignment and collects $\kappa$ deviation in a row. If a synchronous move is found, the advanced cost function is applied to every collected deviation and then the list is emptied, lines 12 and 13. After the loop, the cost of this alignment is calculated, through a loop collecting the cost for each move, and returned. Note that, this algorithm can be applied in an online and offline setting. For the offline setting, the alignment cost can be generated for the complete trace. In an online setting, standard online conformance checking is used [22], which is calculated incrementally. Algorithm 1 is applied every time a new event is detected on the trace the event relates to.

### C. Data Element Acquisition

To use data elements of an event in a process execution log, a distinction of the types of the data elements has to be made, since not every data element type can be used.

**Numerical Data**: Numerical data elements provide a numerical value which is continuous, can be compared to other numerical values and has an ordering. In the process model, for a numerical data element, an interval can be stored to check if the data element in a trace is within a certain range. This interval can be calculated for example using the interquartile range (`IQR`), which equals the difference between the third and first quartile. The lower bound usually then equals the first quartile minus 1.5 times the IQR, while the upper bound equals the third quartile plus 1.5 times the IQR.

In Fig. 5, a small example can be seen for numerical values impacting the moving costs in an alignment. The process model shows the only possible sequence is (`A`,`B`,`C`). Trace `t` shows a possible alignment, where a deviation is seen in the log, since event `B` is missing. We are using 1 for a our basic per-deviation cost penalty, allow one missing event and weigh each data element equally. Since 2 out of 3 data elements at event `C` yield acceptable values and only one deviation is used in this alignment before a synchronous move, it seems plausible that event `B` has taken place and a potential error has occurred while logging. Therefore the cost of this alignment drops to 0.33 instead of 1.

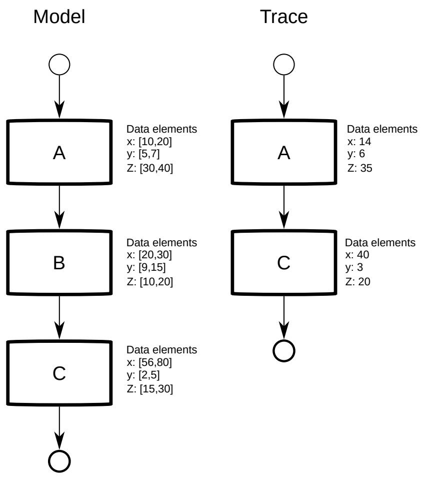


Fig. 5. Example for Numerical Values. A range classifying acceptable values

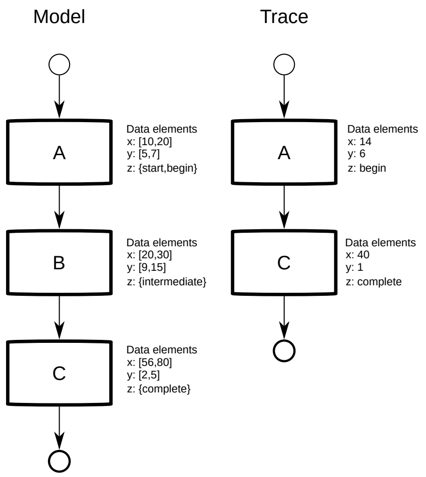


Fig. 6. Example for Categorical Values

**Categorical Data**: This data element type represents data elements with a fixed number of possible values, i.e., day of the week. The XES format allows to relate a data element to a specific extension, like `lifecycle`[4], thus enabling us to distinguish between categorical data elements and arbitrary data elements. To see if a categorical data element contains a correct value, a set of possible values for this data element, is added to an event in the process model. The possible values are values that occurred as value for this data element a certain amount of times, e.g, 30%.

In Fig. 6, again a process model with only one possible sequence, (`A`,`B`,`C`), is shown, but with a different set of relevant data elements attached to event `C`, one of them categorical and two numerical. Trace `t` features a possible alignment with a deviation for event `B`. Again, we assume a basic per-deviation cost of 1, allowing one missing event and weighing each data element equally. Both numerical data elements are not within the corresponding interval, but the categorical data element is one of the 3 possible values, therefore the alignment cost drops to 0.66 instead of 1.

**Time Sequence Data**: In [9] by Dunkl et al., 3 approaches to relate a time sequnce of a sensor event stream to a process model instance are introduced. The first one recording the sensor event stream separately and matching them with the process execution log afterwards. The second one, storing the time sequence in a data element and the third one splitting the sensor event stream into recurring events with a single value into the process execution log. Since we are interested in the time sequence of a data element from one specific event to another, we are using the first approach and slice the time sequence between two specific events for the conformance checking.

In Fig. 7, the process model is enriched with the average time sequences between two events. To compare the time sequence of a process model and the time sequence of a trace between two events, the corresponding time sequences are concatenated together, i.e., for the sequence (`A`,`B`,`C`), the average time sequences between `A` and `B` and, `B` and `C` are concatenated and compared to the time sequence from the trace. To compare time sequences, `Dynamic Time Warping` (DTW) is used [5]. DTW can cope with time sequences of different lenghts and calculates an alignment between both time sequences and the cost for this alignment. If the costs are below a user specific threshold, the costs for a deviation are reduced. To find the average time sequence is calculated using `DTW Barycenter Averaging` (DBA) [14]. The example in Fig. 7, using a basic per-deviation cost of 1, allowing one missing event and weighing each data element equally, shows that in event `C`, both numerical data elements are within the interval of the model and that the distance between the time sequence of data element `x` from the process model and the time sequence of data element `x` from the trace equals 1.73, which is below the threshold of 5, therefore reducing the cost of this alignment to 0.

Other data element types, e.g., arbitrary strings, are currently not supported for this method, since no relation between the data element values is known. Data element types representing a hierarchical structure, like an organizational chart, are currently being assessed and are marked for future work.

**Offline and Online:**

A process history allows us to discover process models online, so at runtime. The information for all relevant data element types can be gathered at runtime, while time sequences from a sensor event stream can be stored online as well. Therefore the information for the advanced cost function differs, that only the currently available and processed data from already executed events out of the event stream can be taken into account. Calculating the alignment of a trace to a process model online uses prefix alignments [22], which are explained in Sec. V. The offline approach gathers the information on data element intervals, sequences and sets usually from a training set out of a process execution log without the need for a process history.

In Sect. IV, the conformance checking method is evaluated

[4] http://www.xes-standard.org/lifecycle.xesext

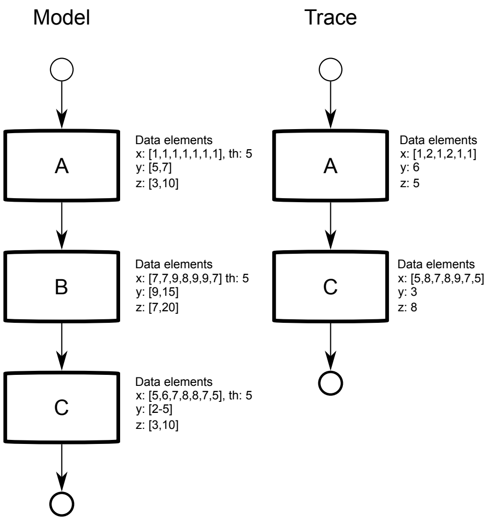


Fig. 7. Example for Time Sequence Values. The average time sequence and the maximal allowed distance to it are stored

using a real word data set from the manufacturing domain and an artificial data set to cover every possible data element type.

## IV. Evaluation

The function and algorithm, elaborated in Sect. III are prototypically implemented and evaluated based on an artificial and real-world log from the manufacturing domain. At the end of the section, the results are discussed.

The evaluation is twofold. The artificial example covers numeric and categorical data elements, while the real-world log is providing time sequence data for one data element.

### *A. Artificial Example*

A test set[5] consisting of 100 traces has been generated following the process model in Fig. 3 and data elements. The complete process model with information on the accepted values for the data elements can be seen in Fig. 8. For the `heart rate` and `blood pressure`, an interval is presented, while for the `status` a set of acceptable values is given.

As mentioned earlier, there are only two combinations for a trace that corresponds perfectly to the process model, i.e., either (`Check General State of Health`, `Administer Drugs`, `Check Blood Pressure`, `Check Heart Rate`) or (`Check General State of Health`, `Administer Drugs` `Check Heart Rate`, `Check Blood Pressure`). From the data set the following acceptable values have been gathered.

- A set consisting of `NOK` for status in `Check General State of Health`.
- A set consisting of `NOK` for status and the values ranging from 61 to 101 for `measurement` in `Check Heart Rate`.
- A set consisting of `NOK` for status and the values ranging from 85.375 to 134.375 for `measurement` in `Check Blood Pressure`.
- A set consisting of `NOK` for status in `Administer Drugs`.

Figure 8 shows the result of Alg. 1 using the process model, $\kappa$ equaling 1 and the sequences shown in the figure. Each event only contains acceptable values.

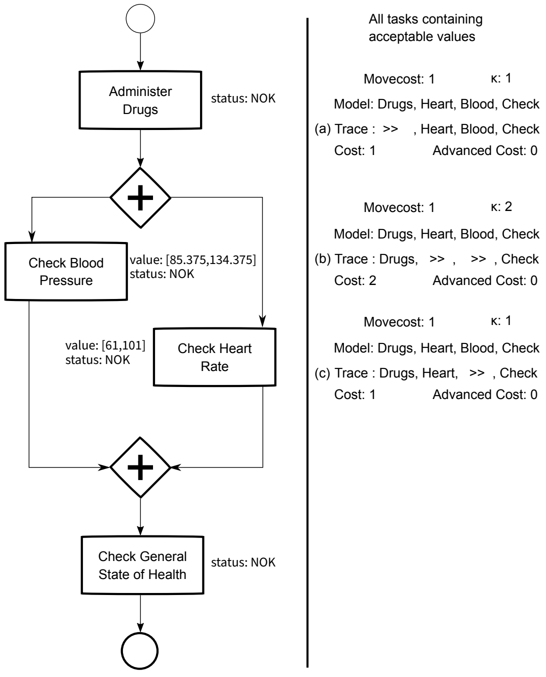

Administer Drugs
status: NOK
Check Blood Pressure
value: [85.375,134.375]
status: NOK
value: [61,101]
status: NOK
Check Heart Rate
Check General State of Health
status: NOK
All tasks containing acceptable values
Movecost: 1
κ: 1
Model: Drugs, Heart, Blood, Check
(a) Trace : >> , Heart, Blood, Check
Cost: 1
Advanced Cost: 0
Movecost: 1
κ: 2
Model: Drugs, Heart, Blood, Check
(b) Trace : Drugs, >> , >> , Check
Cost: 2
Advanced Cost: 0
Movecost: 1
κ: 1
Model: Drugs, Heart, Blood, Check
(c) Trace : Drugs, Heart, >> , Check
Cost: 1
Advanced Cost: 0

Fig. 8. Artificial Example

Sequence `a`, (`Check Heart Rate`, `Check Blood Pressure`, `Check General State of Health`), features a deviation, since `Administer Drugs` is not present in the sequence. Using the standard cost function the cost of the alignment would be 1, but since all the values are acceptable for the following event, the cost is reduced to 0 and an error in the logging is likely, since the heart rate and blood pressure show correct values.

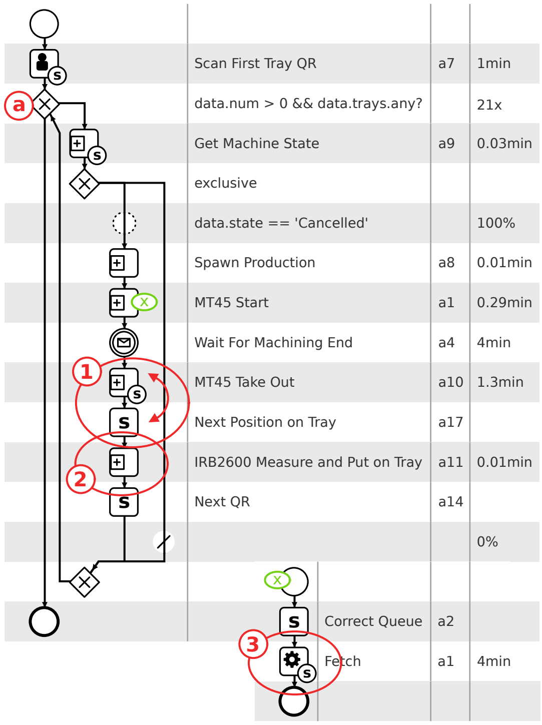
Sequence `b`, (`Administer Drugs`, `Check General State of Health`), features a sequence with 2 deviations in succession. With $\kappa$ set to 1, the first missing event, `Administer Drugs`, is not altered by the advanced cost function. This can be considered by changing the $\kappa$ to 2, which reduces the cost of the alignment to 0.

Sequence `c`, (`Administer Drugs`, `Check Heart Rate`, `Check General State of Health`), features an interesting case, since again, the cost of the alignment can be reduced from 1 to 0, since all values of `Administer Drugs` are acceptable, but since the data element is usually constant over the first 2 events of this process, it can be argued, that this is not caused by an error in the log and the event indeed, never has been executed. This shows that the selection

[5] http://gruppe.wst.univie.ac.at/data/cc/dataset.zip

of relevant data elements is important and has a great impact on the results.

### B. Real World Example



| | | |
|---|---|---|
| Scan First Tray QR | a7 | 1min |
| data.num > 0 && data.trays.any? | | 21x |
| Get Machine State | a9 | 0.03min |
| exclusive | | |
| data.state == 'Cancelled' | | 100% |
| Spawn Production | a8 | 0.01min |
| MT45 Start | a1 | 0.29min |
| Wait For Machining End | a4 | 4min |
| MT45 Take Out | a10 | 1.3min |
| Next Position on Tray | a17 | |
| IRB2600 Measure and Put on Tray | a11 | 0.01min |
| Next QR | a14 | |
| | | 0% |
| Correct Queue | a2 | |
| Fetch | a1 | 4min |

Fig. 9. Manufacturing Example. The 'S' in the modeled tasks, shows that a script is executed by the process execution engine after a task has been completed.

The example shown in Fig. 9 is from the manufacturing domain. The process model described in BPMN[6] is depicted without labels. The labels and decision conditions can be seen in the second column, followed by the task IDs in the third column, and the estimated times (and loop iterations) in the last column.

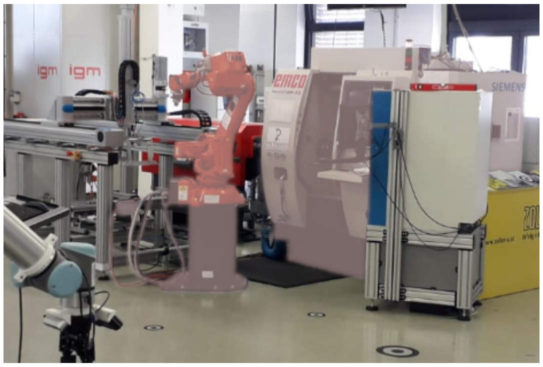


Fig. 10. Shop Floor: Robot, Lathe, Micrometer, Stock

The whole process is executed using the cloud process execution engine (CPEE) [13] on the shop floor (see Fig. 10) of a company specialized on prototyping and small production batches.

Fig. 11. GV12 Part [21]

The process deals with the machining of a part for a gas turbine (see Fig. 11), and models the interplay of three machines and a stock area. MT45 is a lathe by the company EMCO with produces the parts, IRB2600 is a robot by ABB which extracts the machined parts from the lathe, moves it through a high-speed 2D optical micrometer by Keyence in order to get some production quality data. Afterwards the IRB2600 puts each part onto an individual trays (see Fig. 12) in the stocking area. Each tray is labeled with a QR code, it is assumed that the trays are lined up sequentially in the stocking area, so that only the first tray has to be scanned, the QR code for each subsequent tray can then be calculated.

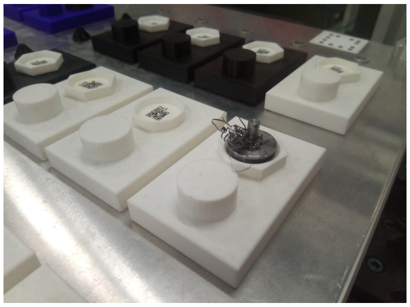


Fig. 12. Trays in Stock Area [21]

The process consists mostly of sub-processes which encapsulate the details of dealing with heterogeneous machine interfaces and operational safety (i.e. while the robot operates, no humans are allowed in the vicinity of the robot). The process is subject to frequent ad-hoc repair actions, as humans frequently stroll into the safety area protected by Pilz light barriers. Whenever that happens, the robot goes into an emergency stop state, that has to be acknowledged by a human. Sometimes the robot is in a position the requires manual position change by an operator in order to avoid damaging equipment, which means some steps in the process model have to be skipped or repeated.

The second process shown in Fig. 9 is part of "MT45 Start", and deals with collecting a stream of data from 14 different sensors in the lathe while the part is produced.

When looking at the loop ⓐ in Fig. 9, out of 21 iterations, 19 had only synchronous moves. The alignments of the other 2 traces are shown in Tab. I and II. $\kappa$ is set to 1 and the base deviation cost is 1. The first alignment features the swap of two events, ①, while the other one

[6] http://www.bpmn.org/

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | Get | Spawn | MT45 Start | Wait | Next | Take out | >> | IRB2600 | QR |
| Trace | Get | Spawn | MT45 Start | Wait | >> | Take out | Next | IRB2600 | QR |

TABLE I
ALIGNMENT OF THE FIRST DEVIATION

the missing of an event, ②. The deviation cost for the first alignment is 2, but in `GV12 MT45 Take Out` and `Next Position on Tray` the data element `tray positions` is present. The range for all 3 values for `tray_positions` equals to [-362.87,437.13], [-382,21,417.79], [220,220]. The values of the trace are in the range of [-112.87,-187.13], [-182.21,217.79],[220,220], which are acceptable. For `GV12 IRB2600 Measure and Put on Tray`, the second deviation, features similar acceptable values, [-212.87, 187.13], [-382.21, 417.79], [220.0, 220.0] are the acceptable values for the data element `value`. The values of the trace -112.87, -182.21 and 220 are acceptable, hence the conformance deviation cost is reduced to 0.

For ②, only one deviation is witnessed. The acceptable values for `Next QR` are [-362.87, 437.13], [-382.21, 417.79] and [220.0, 220.0]. Since the values for the deviated trace are in between [-112.87,187.13],[-182.21,217,79] and [220, 220], the conformance deviation cost is reduced to 0.

In Fig 13 the average time sequence for one sensor can be seen as a red line, ③. The measurements are done in the event `Fetch`, which is done in a loop. The data set was split into a training set consisting of 10 out of the 19 traces that featured traces with a correct outcome, i.e., the part has been correctly built.

A sensor provides time sequence data, so the alignment cost can be reduced if the distance between the time sequence of the model and of the deviated trace is below a certain threshold, here 29.12. This threshold was set by a domain expert.

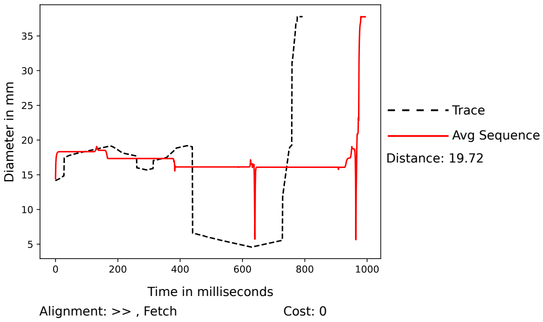


Fig. 13. Time Sequences of Manufacturing Example. Red line is the average time sequence, dashed from one trace. The cost is reduced because the distance is below the threshold of 29.12

Because all traces out of the data set comply to the process model, the first event of the process `Correct Queue` has been randomly removed out of 20 traces.

Figure 13 shows the result of one trace [7] using the advanced cost function. The cost of the alignment can be reduced to 0, because the distance between both time sequences is 19.72 below the threshold of 29.12. This threshold is set by a domain expert.

The 17 traces without missing events, have been aligned perfectly with an alignment cost of 0. Out of the 20 traces, 8 traces resulted in an accepted outcome of the process, while the other 12 traces yielded faulty parts. Out of the 8 traces, the alignment cost of 4 traces has been correctly reduced to 0, since even though the `Fetch` event is missing, the time sequence is below the threshold and the part is correct.

Out of the 12 traces with faulty parts, no alignment costs have been reduced at all, since the time sequence difference was to big. This leads to believe that the error is in the execution and not in the logging, therefore the alignment cost using the standard function is correct.

## C. Discussion

The results of the artificial and the real world example showed interesting results. The best results are achieved using data elements that always yield the same values in every process instance. For example, in the manufacturing domain, the dimensions of the produced part are always the same, thus the interval for this data elements can be calculated easily. The same can be observed while using time sequence data elements. Since the time sequences are fairly similar of all process instances. On the other hand, i.e., if the values for the data elements vary and do not follow a pattern, the advanced cost function cannot be applied. For example in the finance domain in a loan approval process, the amount of the loan can vary greatly and the different instances are not related.

This leads to an important aspect of the advanced cost function, the selection of relevant data elements of the events. This can be seen in the artificial example.

As mentioned before, data element that vary greatly are problematic, since acceptable values would contain a great range. This would result in a reduced alignment cost, since almost all values are acceptable for such a data element even though it is faulty. Therefore experts have to select viable data elements to apply the advanced cost function onto.

The value of a data element at a certain amount of time, always leaves some room for interpretation. Time sequence on the other hand shows the full history of a data element. For example, the blood pressure of a patient should be elevated for a certain time span instead of one measurement. It can be seen that the advanced cost function on the real world data set assigns lower alignment cost values for good parts and does not reduce the alignment cost for faulty parts for 50% of the traces. This leads to the assumption, that the costs be calculated best using time sequence data.

[7] 79917b2a-5bac-4074-8690-4e4e5cdf0b8f

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model | Get | Spawn | MT45 Start | Wait | Take out | Next | IRB2600 | QR |
| Trace | Get | Spawn | MT45 Start | Wait | Take out | Next | >> | QR |

TABLE II
ALIGNMENT OF THE SECOND DEVIATION

| | Trace with Correct Parts | Trace with Faulty Part |
|---|---|---|
| Traces with reduced costs | 50% | 0% |
| Traces with no reduced costs | 50% | 100% |

TABLE III
RESULTS ALG. 1

## V. RELATED WORK

Several algorithms for offline process conformance checking exist [3]. In the beginning, process conformance checking was performed doing a replay of process model using tokens, typically a Petri Net. After the replay the fitness of a process instance was then calculated using the missing and remaining tokens that were still in the Petri net [17], [2]. Nowadays alignments [4] are used to calculate the fitness of a process instance by assigning a cost to missing or additional events in the event log and or the process model. Calculating these alignments is an immense computational task, since lots of variations have to be considered finding the best alignment. To tackle this problem, an approximation of the conformance value is introduced by [18].

Usually this methods are done offline, i.e., ex-post. Therefore the complete process has been already executed before any results are yielded. There currently promising approaches that aim at eliminating this drawback and perform conformance checking at run-time using an event stream [7] instead of an event log. In such an online approach [22], prefix alignments are calculated incrementally, since the future execution path of a process instance is not available all the time.

These approaches usually are always focused on the control flow of a process instance. To match an event of a process instance to a process model, an identifier is needed. Most of the currently available process mining techniques therefore match events with their labels, which is a potential source of errors [15].

To negate this, [16], are not using an event log at all, but monitor the data elements related to a process instance. The change of data elements are logged as a sequence of measurements, a time series. A workflow net is then enriched by these time sequences. Their approach aims at the differences between this workflow net and the log of time sequences. In comparison to presented work, the information of an event log is not used anymore. The differences between the time sequences is also not calculated on the time sequence as a whole, like the difference between two time sequences, i.e., using Dynamic Time Warping [5], but instead detects if the data elements are within certain intervals and increase or decrease at the right time and ist done ex-post.

Another interesting aspect of current techniques in conformance checking is that usually the likelihood of observed events is not taken into consideration. Stochastic Conformance Checking [12] tackles this question. It translates event logs into a stochastic language and stochastic petri nets, which contains for probability of a specific transition firing. Stochastic Conformance Checking then calculates the conformance of an event log by analyzing the distribution of the stochastic petri net and the event log using the Earth Movers' Distance. This distance represents the cost for transforming a pile of earth,i.e. a distribution, to another one. This approach is not considering data elements attached to an event and is done ex-post.

## VI. CONCLUSION

We introduced a novel approach to calculate the deviation cost in an alignment for conformance checking in an online and offline way. The approach features an advanced cost function to alter the deviation costs after an alignment using the standard cost function is calculated. The data elements that are attached to events are used to distinguish errors in the logging and security breaches, giving a more detailed view on the results of standard conformance checking.

The required data to make this distinction is gathered from an process execution log. The approach is evaluated on an artificial and a real world example. The results are promising. For future work, working with hierarchical data elements, like an organizational chart, is planned.

## ACKNOWLEDGMENT

This work has been partly funded by the Austrian Research Promotion Agency (FFG) via the "Austrian Competence Center for Digital Production" (CDP) under the contract number 854187. This work has been supported by the Pilot Factory Industry 4.0, Seestadtstrasse 27, Vienna, Austria.

## REFERENCES

[1] IEEE standard for extensible event stream (XES) for achieving interoperability in event logs and event streams. IEEE Std 1849-2016 pp. 1–50 (Nov 2016)

[2] Van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2(2), 182–192 (2012)

[3] van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)

[4] Adriansyah, A.: Aligning observed and modeled behavior. Ph.D. thesis, Department of Mathematics and Computer Science (2014)

[5] Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD workshop. vol. 10, pp. 359–370. Seattle, WA (1994)

[6] Burattin, A., Carmona, J.: A framework for online conformance checking. In: International Conference on Business Process Management. pp. 165–177. Springer (2017)

[7] Burattin, A., Sperduti, A., van der Aalst, W.M.: Control-flow discovery from event streams. In: 2014 IEEE Congress on Evolutionary Computation (CEC). pp. 2420–2427. IEEE (2014)

[8] Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018), https://doi.org/10.1007/978-3-319-99414-7

[9] Dunkl, R., Rinderle-Ma, S., Grossmann, W., Fröschl, K.A.: A method for analyzing time series data in process mining: application and extension of decision point analysis. In: CAiSE Forum. pp. 68–84 (2014)

[10] Grossmann, W., Rinderle-Ma, S.: Fundamentals of Business intelligence. Springer (2015)

[11] Kaes, G., Mangler, J., Stertz, F., Vigne, R., Rinderle-Ma, S.: Acaplan-adaptive care planning (2015)

[12] Leemans, S.J., Syring, A.F., van der Aalst, W.M.: Earth movers stochastic conformance checking. In: International Conference on Business Process Management. pp. 127–143. Springer (2019)

[13] Mangler, J., Rinderle-Ma, S.: Cpee-cloud process exection engine (2014)

[14] Petitjean, F., Ketterlin, A., Gançarski, P.: A global averaging method for dynamic time warping, with applications to clustering. Pattern Recognition 44(3), 678–693 (2011)

[15] Rinderle-Ma, S., Reichert, M., Jurisch, M.: On utilizing web service equivalence for supporting the composition life cycle. International Journal of Web Services Research (IJWSR) 8(1), 41–67 (2011)

[16] Rodriguez-Fernandez, V., Trzcionkowska, A., Gonzalez-Pardo, A., Brzychczy, E., Nalepa, G.J., Camacho, D.: Conformance checking for time series-aware processes. IEEE Transactions on Industrial Informatics pp. 1–1 (2020)

[17] Rozinat, A., Van der Aalst, W.M.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. 33(1), 64–95 (2008)

[18] Sani, M.F., van Zelst, S.J., van der Aalst, W.M.: Conformance checking approximation using subset selection and edit distance. In: International Conference on Advanced Information Systems Engineering. pp. 234–251. Springer (2020)

[19] Stertz, F., Rinderle-Ma, S.: Process histories – detecting and representing concept drifts based on event streams. In: Cooperative Information Systems. pp. 318–335 (2018)

[20] Stertz, F., Rinderle-Ma, S.: Detecting and identifying data drifts in process event streams based on process histories. In: CAiSE Forum. pp. 240–252 (2019)

[21] Stertz, F., Rinderle-Ma, S., Mangler, J.: Analyzing process concept drifts based on sensor event streams during runtime. In: to appear in Business Process Management, 18th International Conference, BPM 2020. Springer International Publishing (2020)

[22] van Zelst, S.J., Bolt, A., Hassani, M., van Dongen, B.F., van der Aalst, W.M.: Online conformance checking: relating event streams to process models using prefix-alignments. Data Science and Analytics pp. 1–16 (2017)