Interactive Drift Visualization in Sensor Data Streams for Explainable Process Outcome Prediction

Matthias Ehrendorfer$^{1[0000-0002-7739-9123]}$, Jennifer Hebstreit^{1}, Juergen Mangler$^{1[0000-0002-6332-5801]}$, and Stefanie Rinderle-Ma$^{1[0000-0001-5656-6108]}$

Technical University of Munich, Germany; TUM School of Computation, Information, and Technology
{matthias.ehrendorfer, jennifer.hebstreit, juergen.mangler, stefanie.rinderle-ma}@tum.de

Abstract. In real-world process scenarios such as manufacturing and logistics, the process outcome is frequently predicted by IoT sensor data streams and their drifts, e.g., the quality of a product can be affected by the temperature during the production process. In particular, drifts can explain variations in the process outcome, and hence, their early detection can support the definition of mitigation actions, e.g., canceling the production of probably low quality products. As in most cases, humans have to define such actions, it is crucial to support them in making decisions in the context of outcome prediction. Hence, this paper aims to support the interactive visualization of drifts in specific points of sensor data streams to show the development of critical sensor measurement points over different traces. Furthermore, these critical points can be identified based on drifts between distinct groups of traces. Being able to visualize drifts and identify critical points enables (early) outcome prediction, ranging from the analysis of drifts at single points or at specific timestamps in a trace to the investigation of average time series of traces representing different outcomes, e.g., OK/NOK. Three different methods to derive and visualize drift points are presented and evaluated based on a prototypical implementation and a survey with users.

Keywords: BPM and IoT · Sensor Data Streams · Process Outcome Prediction · Drift Visualization · Drift Analysis.

1 Introduction

Process models provide a blueprint on how and in which order the tasks of a process should be performed. However, when process models are enacted repeatedly, over time, there might be changes in the order of the tasks or how they are carried out. Such changes are called drifts which might ultimately lead to an adaptation of the process model [1]. Naturally, such drifts can happen not only in the control-flow perspective of the process (as described above) but also in other perspectives, such as the organizational perspective, if roles carrying out the tasks change or the data perspective, when data elements obtained during the process (e.g., amount of a loan or age of an applicant) start to shift over the course of multiple enacted processes [11].

Another source for drifts to occur lies in the sensor data streams that are recorded during the execution of different traces of a process [12]. Figure 1 presents an example, how a sensor data stream of a temperature recording could look like for two traces. The drift between two traces here signifies a potential thermal runoff affecting the process outcome. Its detection would allow users to rectify the situation by increasing the machine downtime, to achieve a stable production procedure. *Visual drift detection* seems easy for two sensor data streams, but becomes challenging for multiple sensor data streams with multiple drifts between the traces. Hence, this work asks **how users can be supported in visually detecting drifts in sensor data streams and assessing their effects on the process outcome.**

In general, the detection of drifts in sensor data streams is important because it provides information on how a specific process instance progresses and allows to react if undesired behavior occurs. This can be done by either correcting the situation for the currently analyzed instance or by letting a domain expert decide on how and if such deviations/drifts could be avoided or handled in the future, which leads to an improved process model.

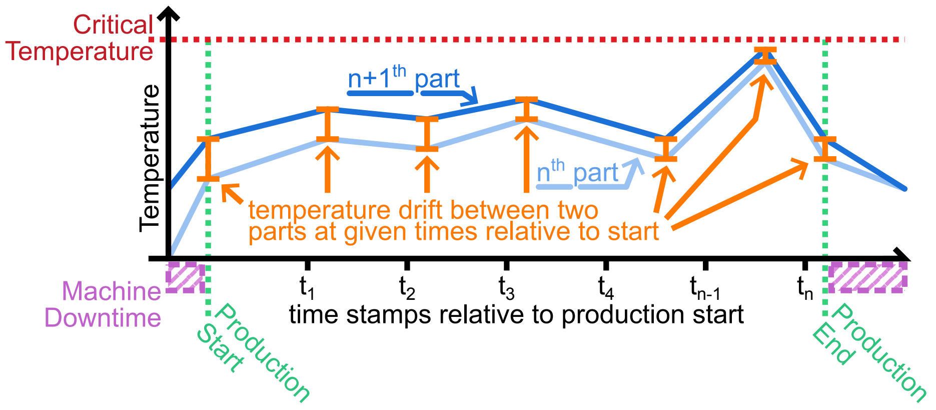


Fig. 1: Temperature Drift in a Production Process

The contribution of this paper is three-fold: **Contribution C1:** We propose an approach to visualize drifts for multiple traces in an easy-to-perceive way. **Contribution C2:** By using information about drifts between traces, we show how outcome prediction by considering drifts in different points of the sensor data streams, becomes possible. This method of outcome prediction has the advantage that it perfectly explains which points led to the decision and how the sensor data drifts in these points behave compared to other traces. **Contribution C3:** We evaluate the approach by implementation and conducting a survey with experts in the field.

The remainder of the paper is structured as follows: Section 2 discusses related work, Section 3 describes the presented approach, Section 4 provides a survey and the results for outcome prediction performed with the proposed methods, Section 5 discusses limitations, and Section 6 concludes the paper.

2 Related Work

The visualization of processes and their execution plays a crucial role in understanding and managing complex data flows and process behaviors. Business process management often employs BPMN to model processes graphically, enabling stakeholders to comprehend workflows comprehensively and identify inefficiencies. Utilizing process models can significantly enhance decision-making, potentially leading to improved operational efficiency and increased revenue [2]. On the execution side sensor event streams can play a crucial part in explaining the root cause of deviations from expected process behaviors [12]. To visualize these deviations this work aims to provide an interactive visualization framework for drifts in sensor event streams, enhancing the understanding of process outcomes and enabling more informed decision-making. [5] highlights the value of visualizing data from manufacturing execution systems, drawing special attention to how interactive visualizations empower users to selectively focus on data points relevant to their decision-making needs. One approach in ubiquitous computing visualizes time-series sensor data to aid in just-in-time adaptive intervention design [10]. However, the mentioned works fall short of addressing concept drifts and the challenges posed by concept drifts in process execution.

Concept drift and data drift affect the accuracy and reliability of numerous real-world applications exposed to dynamic environments [14] [3]. Concept drifts are changes in the control-flow of a process, which are detected through analyzing process execution logs [11]. In existing work in machine learning, data drift is defined as the discrepancies between the data set initially used to train a machine learning model and the data subsequently collected by the model during real-world application [3]. This paper follows the definition of data drift as defined by [11]: it is the change in process data, e.g., when machine errors change machining parameters during production in manufacturing processes. Due to the different types of input data and detection methods employed, many different visualizations of concept drifts were developed in existing work across different domains. Extensive research exists on the visualization of drifts between process models [9], event logs [15], and multidimensional problem spaces [7]. Concept drifts have also been visualized using feature importance on streaming data [8]. Other fields, such as cloud computing, produced works on cluster-based data drift visualizations of CPU and memory resource usage [4]. However, the mentioned works do not examine external data such as IoT sensor data or visualize drifts over distinct groups of traces.

3 Approach

Three methods to visualize the drift between a specified point in a sensor data stream and another sensor data stream are presented in Sect. 3.1. Furthermore, an approach to automatically determine interesting points in sensor data streams which should be observed with the aforementioned methods is presented in Sect. 3.2.

3.1 Visualization Approaches

Data Preparation Log files of process instances contain sensor event streams that capture data from external sensors monitoring the physical conditions under which the process is executed. Each data point corresponds to a single time-stamped measurement from the sensor, which, compiled, constitute time sequence data. This data needs to be resampled into discrete and equidistant time series data like in [12] to tackle the problem of unevenly distributed time sequences and obtain an even time scale over all available time sequences. Afterwards, the most recent time series are considered, and outliers (i.e., whole time series that are too long/short) within this predetermined window are excluded based on the criteria *"sequences with a duration shorter than the first quartile minus 1.5 times the IQR (Interquartile Range) or with a duration greater than the third quartile plus 1.5 times the IQR, similar to boxplots. The IQR is calculated here between third and first quartile."* [12]. This outlier detection method is adapted from [12] and modified to simultaneously detect outlier traces from all available time sequence data instead of performing the method trace by trace. Moreover, the method works only on already completed time series (because start and end events must be known) and when outliers are detected, the whole time series is excluded.

After outlier removal, traces are grouped to allow more efficient drift visualization with a large number of time series. For each group an average time series (ATS) can be calculated as described in [12], by utilizing the DTW Barycenter Averaging (DBA) [6] algorithm as distance measure. Averaging sequences using DBA involves an iterative refinement process that aims to minimize the squared distance (DTW) to averaged sequences, even if the initial average sequence is arbitrary [6]. As a result, the computation time for this technique is quadratic, as a DTW matrix must be generated for each iteration [12]. In this paper, two approaches for grouping the traces are used. In the first one, traces are split up based on their time of occurrence into fix-sized groups to detect drifts that happen over time, e.g., because of tool wear. In the second one, the traces are grouped based on their process outcome, i.e., whether the part produced in the process is OK or not OK.

Data Visualization The three novel drift visualization methods presented comprise 90 degrees angle (*90Deg*), shortest distance (*SD*), and same timestamp (*ST*) drift visualization. They allow to compare a specified point in one time series (in the following called **analyzed time series**) to another time series (in the following called **target time series**). Both time series can either be a time series from an individual trace or an average time series obtained from multiple traces. Figure 2 shows an illustrative example with a time series of sensor measurements from an individual trace (i.e., black line) and average time series (i.e., red, blue, and green lines), each being derived from sensor measurement of multiple consecutive traces. Drifts between time series are shown with orange, pink, and purple arrows at different points in time (green dots). Techniques that scale the lengths of the lines with a user-selected factor and insert offsets to add space between overlaps (see pale purple line between pink and purple lines) are used to prevent overlapping lines and barely perceivable differences. Figure 2

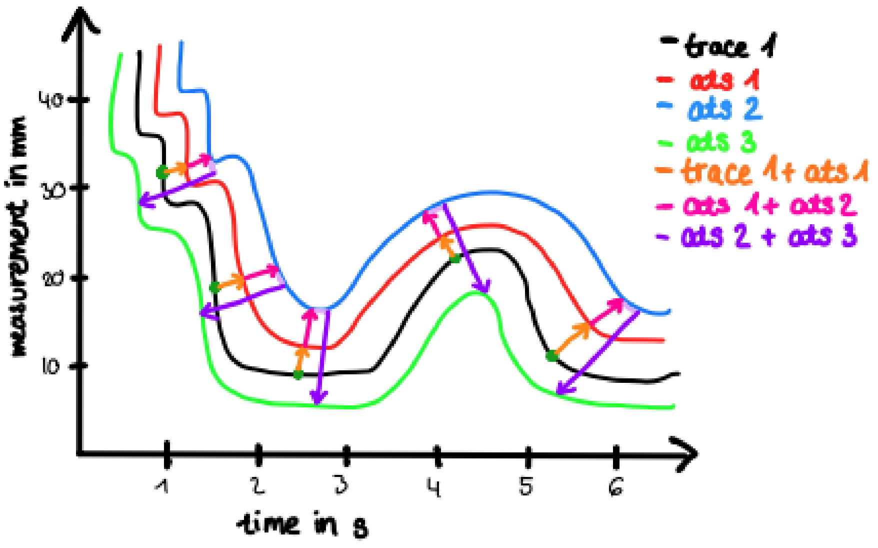
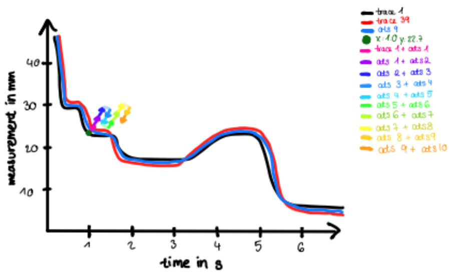


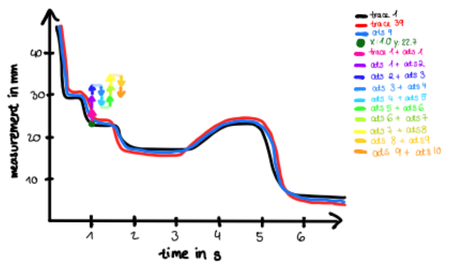
Fig. 2: Underlying Idea Behind the Drift Visualization Techniques

shows that calculating drifts between time series depends on which point on the target time series is used to measure the distance to the analyzed time series (i.e., distance between the defined point on the analyzed time series and the one chosen on the target time series). Different methods for finding the point on the target time series are possible:

- ***90Deg*** **Drift Visualization Method:** This method involves calculating a line perpendicular to the point on the analyzed time series by considering the slope of its neighboring points. Then, the nearest intersection point on the target time series is found, and the distance between the starting point on the analyzed time series and this point represents the drift. Algorithm 1 outlines the pseudo code for this method. It requires a time series, a list of average time series, a specific timestamp to be analyzed (`point_x`), a scaling factor, values for segment spacing, and neighbor point distance. First, lists for storing line segments and process execution are initialized and then the sensor measurement corresponding to `point_x` is determined (`get_point` function) and used as the starting point of the drift calculations. The starting point, alongside `distance` and the first average time series, is used to calculate intersection points with the next time series by finding neighboring points to construct the slope of a perpendicular line that crosses the next average time series. The `identify_intersection` function selects the closest intersection point to `p1`, or the closest point on the average time series if no intersection exists. The x-value of `p1` is logged in `proc_exec`. The method iterates through subsequent average time series to calculate new intersection points, which are then used as bases for the next calculations. Finally, the line segments are scaled and offset for improved visualization, returning the (modified) line segments and process execution identifiers. Figure 3a depicts the expected behavior of this visualization.



(a) *90Deg* Drift Visualization



(b) *ST* Drift Visualization

Fig. 3: Behavior of 90Deg, ST Drift Visualizations

– ***SD* Drift Visualization Method**: This method portrays drifts as the minimum distance between a point on the analyzed time series and the target time series. At the specified point, the smallest distance to a point on the target time series must be found. Algorithm 2 shows pseudo-code for this method. It calculates the shortest distance from the starting point to the next average time series, using the project method to find the minimum distance and then linear interpolation to identify the data point on the average time series. This point and starting point are added to `line_segments`, and its x-value to `proc_exec`. This is done for each average time series, using the last point found as the starting point for the next calculation. Line segments are scaled, and offsets are added to prevent overlap if set by the user. Afterwards, the (updated) line segments and `proc_exec` list are returned.

```
Input: ts, ats_list, point_x, scale_factor, offset, distance
Output: scaled_and_spaced_lengths, line_segments, proc_exec
line_segments, proc_exec = []
p1 = get_point(ts, point_x)
original = ts
proc_exec.append(p1.x)
for ats in ats_list do
    intersections = get_perp_line(original, p1, dist, ats)
    closest_point = identify_intersection(intersections, ats)
    line_segments.append([p1, closest_point])
    proc_exec.append(closest_point.x)
    p1 = closest_point, original = ats
end
scaled_lengths = scale_lengths(line_segments, scale_factor)
scaled_and_spaced_lengths = add_offset(scaled_lengths, offset)
return scaled_and_spaced_lengths, line_segments, proc_exec
```

Algorithm 1: Calculate *90Deg* Drift

```
Input: ts, ats_list, point_x, scale_factor, offset
Output: scaled_and_spaced_lengths, line_segments, proc_exec
line_segments, proc_exec = []
p1 = get_point(ts, point_x)
proc_exec.append(p1.x)
for ats in ats_list do
    closest_point = ats.interpolate(ats.project(p1))
    line_segments.append([p1, closest_point])
    proc_exec.append(closest_point.x)
    p1 = closest_point
end
scaled_lengths = scale_lengths(line_segments, scale_factor)
scaled_and_spaced_lengths = add_offset(scaled_lengths, offset)
return scaled_and_spaced_lengths, line_segments, proc_exec
```

Algorithm 2: Calculate *SD* Drift

– ***ST* Drift Visualization Method:** This method visualizes the drift between a point in the analyzed time series and the target time series by calculating the distances between measurements on these two time series at the same timestamp. Algorithm 3 details the pseudo code: The method uses linear interpolation to find a y-value corresponding to an x-value from the next average time series, forming line segments for drift visualization with the start point. The x-coordinate of this point is recorded for process information retrieval. This interpolation process repeats for each subsequent average time series, with the last intersection point as the new start. The resulting line segments are adjusted for scale and spacing according to the input, and the method returns this information. Figure 3b displays the expected behavior of this method.

```
Input: ts, ats_list, point_x, scale_factor, offset
Output: scaled_and_spaced_lengths, line_segments, proc_exec
line_segments, proc_exec = []
p1 = get_point(ts, point_x)
proc_exec.append(p1.x)
for ats in ats_list do
    ipvy = interpolate(p1.x, ats)
    closest_point = (p1.x, ipvy)
    proc_exec.append(closest_point.x)
    line_segments.append([p1, closest_point])
    p1 = closest_point
end
scaled_lengths = scale_lengths(line_segments, scale_factor)
scaled_and_spaced_lengths = add_offset(scaled_lengths, offset)
return scaled_and_spaced_lengths, line_segments, proc_exec
```

Algorithm 3: Calculate ST Drift

3.2 Point of Interest Detection

The methods described in Sect. 3.1 allow visualization and calculation of drifts in one point of an analyzed time series. However, which **point(s) of interest (POI(s))** are to be observed is not specified and in the following two approaches to find such POIs are proposed:

Point of Interest Detection Based on Individual Traces Detecting POIs for individual traces uses only information from the analyzed trace itself by fitting a curve to the available points after the Data Preparation step (cf. Sect. 3.1). Afterwards, maximal and minimal points (i.e., extreme points), as well as points with a strong change (i.e., inflection points), are used as POIs of the series. Therefore, each individual trace has its own POIs. However, the time series need to already be completely recorded to allow curve fitting.

Point of Interest Detection Based on Analyzing Multiple Traces The detection of POIs based on multiple traces is described in Alg. 4. First, traces have to be split up into two classes based on some characteristic that identifies them (e.g., traces leading to OK vs. not OK parts). Then, for each class, the average time series can be calculated as described in Sect. 3.1. With these two average time series as input for every point in one of the average time series (**ats_A** in the algorithm and the remainder of this section) the distance to the other average time series (**ats_B** in the algorithm and the remainder of this section) is calculated based on one of the three methods described in Sect. 3.1. The results are then separated by points where the distance between ats_A to ats_B is zero or close to zero (i.e., where they intersect) to split the time series into segments where they deviate. Finally, the maximal distance isMaxInSegment in the algorithm, as well as extreme points (calculated by looking at upwards trends of minimal length min_trend_length and then choosing the highest point out of the next points, denoted as isMaxInUpwardsTrend in the algorithm), are used as POIs because in these timestamps ats_A deviates the most from ats_B (under the used distance metric).

```
Input: (average time series) ats_A, ats_B, min_trend_length
Output: pois
segments, pois = []
for point in ats_A do
    if calculate_distance(point,ats_B) ≤ 0.01 then
        segments.append([])
    end
    segments.last.append([point.x, calculate_distance(point,ats_B)])
end
for segment in segments do
    for point in segment do
        if isMaxInSegment(point,segment) OR
          isMaxInUpwardsTrend(point,segment,min_trend_length) then
            pois.append(point)
        end
    end
end
return pois
```

Algorithm 4: Determine POIs Based on Multiple Traces

4 Evaluation

The first part of the evaluation described in Sect. 4.1 assesses the proposed drift visualization methods by conducting a survey comparing them (including their additional features like getting rid of overlapping drifts) to the visualization of raw sensor data streams. The second part described in Sect. 4.2 evaluates the approaches for finding POIs by (1) using the automatically determined POIs to predict outcomes for traces and (2) conducting a survey based on which criteria experts would choose POIs and to which conclusion regarding the outcome of traces they would come based on the chosen POIs.

The online questionnaire created for the evaluation contains closed-ended questions about the visualization methods using a six-point Likert scale (series of statements with which respondents can state their level of agreement [13]) as answer options range from "Strongly Disagree" to "Strongly Agree". According to [13], using a six to seven-point rating scale is recommended. We opted for a six-point scale to prevent neutral answers. Furthermore, open questions were included to get additional insights. The second part of the questionnaire entailed closed-ended questions inducing binary responses (yes/no) with some questions also allowing "undecidable". The questions deal with which points in a trace should be analyzed in-depth and what that means for predicting the outcome of a trace. Additionally, open questions are asked to find out how respondents came up with their answers. The survey was completed by ten experts in the field of business process management (between 2 and 24 years experience) with different areas of expertise reaching from process modelling and process mining to resource allocation and IoT data integration.

The data set used for the questionnaire and evaluation contains logs of producing 37 parts in a manufacturing process. Each part is produced by a machine tool and measured directly afterwards by a fast, but imprecise measuring machine and finally by a slower, but more precise measuring machine. For the evaluation, the sensor data stream of the first measurement, which measures the diameter of the part while moved through the measuring machine, is used to detect drifts between different traces. The second measurement is used to define the outcome (18 parts are faulty while 19 are OK) of a trace.

The survey was performed using Google Forms and the questions are made available on Zenodo [1] together with the results. The code used to create the screenshots for the first part of the questionnaire focusing on the evaluation of drift visualization methods, including a description of how to execute it, is available on GitHub[2]. On two other GitHub repositories [3] [4], the code utilized in the second part of the questionnaire that analyzed traces based on different drift visualization methods and used for the outcome prediction evaluation, can be found. The repositories also contain the used data sets.

4.1 Evaluation of the Proposed Visualization Methods

The conducted survey evaluates if the drift visualization methods presented in Sect. 3.1 are useful in displaying drifts happening in one sensor data stream over several traces and investigates details of the approaches. Therefore, for three different points in a trace, all three proposed methods for drift visualization are applied. The participants are presented with a picture of a specific method applied on a specific point together with a text explaining the parts of the picture. Five questions (described below in more detail) assessing how well the drift visualization works are asked with six answer options (from “Strongly Agree” to “Strongly Disagree”) for each point/method combination.

For subfigures (a)–(e) in Fig. 4, we report on the number of positive responses (i.e., “Strongly Agree”, “Agree”, “Slightly Agree”) to the questions. For the first question “Can you easily identify in which direction measurements have drifted?” (see Fig. 4a) 8 out of 10 responses are positive for all point/method combinations apart from the ST method at timestamp 4.75. The question “Can

[1] `https://doi.org/10.5281/zenodo.11654993`, accessed on 14th June 2024

[2] `https://github.com/jennvheb/ba_drift_visualization`, accessed on 14th June 2024

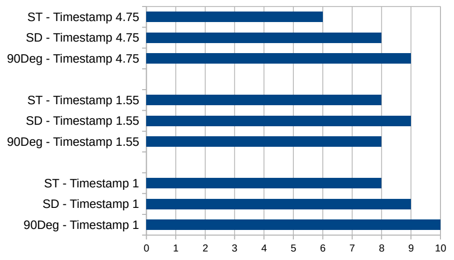
[3] `https://github.com/jennvheb/paper_drift_visualization`, accessed on 14th June 2024

[4] `https://github.com/me33551/drift_visualization_based_outcome_prediction`, accessed on 14th June 2024

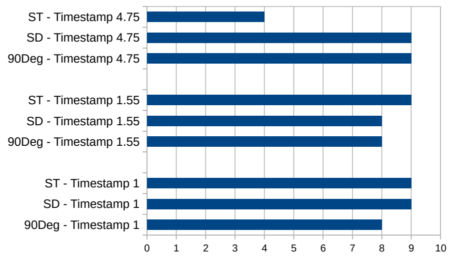
you easily identify the overall amount of drift that occurred?" (see Fig. 4b) also achieves at least 8 positive responses for all combinations except the *ST* method at timestamp 4.75. For question "Can you easily identify individual drift amounts between certain grouped traces (e.g., between traces 6-10 and 11-15)?" (see Fig. 4c) positive responses are lower. Again, *ST* at timestamp 4.75 got the lowest number of positive responses (4), and also for timestamp 1.55, only 5 participants respond positively to the question. Interestingly, the *ST* method at timestamp 1 receives the most positive responses. For the question "Are the details displayed for a certain drift (arrow) useful?" (see Fig. 4d) the *ST* method at timestamp 1 receives 9 positive responses while the same method at timestamp 4.75 receives the least number of positive responses (3). The question "Is the offset helpful in identifying individual drift amounts?" (see Fig. 4e) receives between 7 and 10 positive responses for all combinations apart from the *ST* method at timestamp 4.75. Overall, all participants, apart from one, disagree (i.e., "Strongly Disagree", "Disagree", or "Slightly Disagree") with the statement that "A concept drift is easier to spot with the raw data visualization than with ..." for every method shown (see Fig. 4f). However, disagreement is stronger for the *ST* and *SD* visualization methods compared to the *90Deg* method. Participants' answers in the open questions where they were asked (after seeing each method for a specific timestamp) to sum up the strengths and weaknesses of each method reveal that:

– For timestamp 1 participants agreed that they liked the *ST* method (because it is easy to understand) best. However, one participant also mentioned the *90Deg* method as intuitive.
– For timestamp 1.55 participants describe that using approaches 1 and 2 (i.e., *90Deg* and *SD* visualization method) makes more sense for them when looking at this timestamp compared to the first one because it also captures some kind of "temporal" shift.
– For timestamp 4.75 participants again describe that using approaches 1 and 2 (i.e., *90Deg* and *SD* visualization method) makes more sense for them when looking at this timestamp. Additionally, they describe that the *ST* visualization method does not provide any real insight into the data.
– The *ST* visualization method is perceived as the most native/easiest to understand while the other ones are described as slightly confusing by some participants.
– Some participants state in their answers among different timestamps that offsetting the drifts in the visualizations is useful for them.

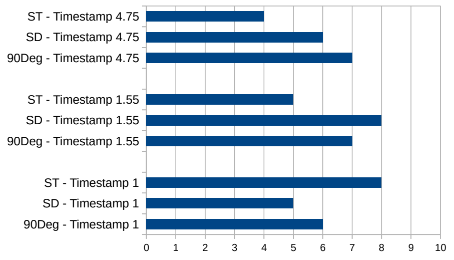
Overall, participants' answers to the open questions described above and the questions summarized in Fig. 4 show that (1) drifts can be spotted easier with the presented approaches than by just looking at the raw data, (2) even inside the same scenario depending on the timestamp that should be analyzed different visualization methods might be useful (3) preventing overlaps of the visualized drifts makes it easier to perceive drifts.



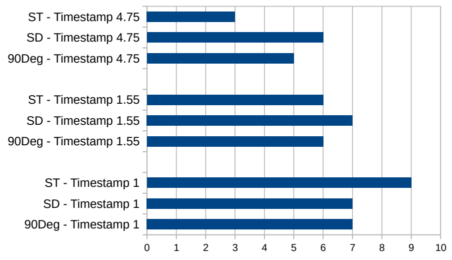
(a) Can you easily identify in which direction measurements have drifted?



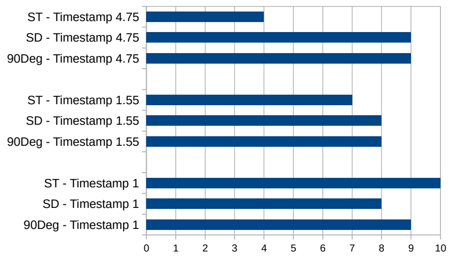
(b) Can you easily identify the overall amount of drift that occurred?



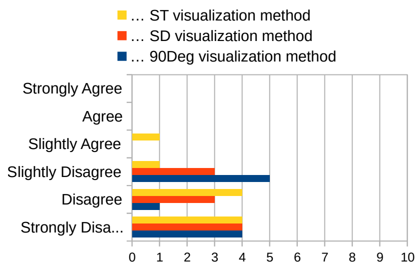
(c) Can you easily identify individual drift amounts between certain grouped traces (e.g., between traces 6-10 and 11-15)?



(d) Are the details displayed for a certain drift (arrow) useful?



(e) Is the offset helpful in identifying individual drift amounts?



(f) A concept drift is easier to spot with the raw data visualization than with ...

Fig. 4: Summarized Survey Results for the Proposed Visualization Methods

4.2 Point of Interest Detection for Outcome Prediction

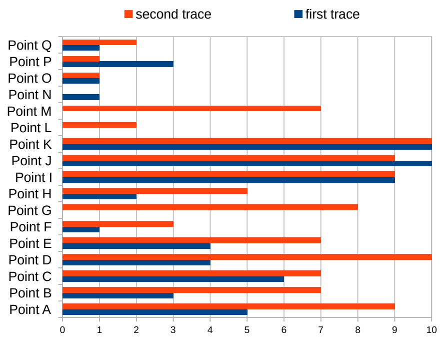
Evaluating the detection of POIs and subsequent outcome prediction is done by applying the techniques described in Sect. 3.2 and using the POIs found to predict the outcome of traces. The trace is predicted to belong to the class to which average time series more POIs of a trace are closer.

The results are presented in Tab. 1 and show in the leftmost column which method was used to obtain POIs (either "individual trace" for the approach described in Sect. 3.2 using individual traces for POI detection or *90Deg*, *SD*, or *ST* for the approach described in Sect. 3.2 based on multiple traces). Additionally, it is important if the distance is measured from a point in the average OK time series to the average not OK time series or the other way around (i.e., which one is ats_A and which one is ats_B in Alg. 4), i.e.,"OK/NOK" or "NOK/OK". One row in the table uses POIs derived with the given method (with min_trend_length = 3) and calculates the distance to the average OK and not OK time series for each POI (i.e., each timestamp) in each trace. The three methods described in Sect. 3.1 are utilized (see the first line of the table) for this distance calculation. As explained above, the number of POIs closer to the OK or not OK average time series is used to predict the produced part quality of each trace. This is reported as the number of correctly classified traces (column labelled "correct"/"c"), number of incorrectly classified traces (column labelled "incorrect"/"i") and number of "undecidable"/"u" traces (e.g., the same number of POIs is closer to the average OK and average not OK time series). It can be seen in Tab.1 that the approaches based on finding points using multiple traces (lines 4–9) are slightly better than the ones performed on each trace individually (line 3). The approaches to find POIs shown in lines 4–9 perform similarly even when using different distance metrics for measuring the distance from POIs to the average OK or not OK traces. Overall, in these approaches 18–21 traces out of 37 are correctly predicted ($\approx$ 48.65%–54.05%) while 4–8 traces remain undecidable ($\approx$ 10.81%–21.62%) and 9–15 traces are incorrectly classified ($\approx$ 24.32%–40.54%).

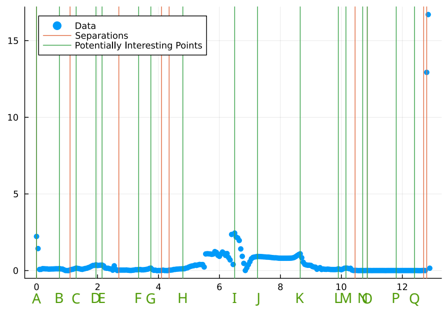
Table 1: Predicted Outcome of Traces Based on Automatic and Manual POIs

	90Deg			SD			ST		
	correct	undecidable	incorrect	c	u	i	c	u	i
individual trace	17	8	12	17	8	12	16	8	13
90Deg OK/NOK	19	4	14	20	4	13	21	4	12
90Deg NOK/OK	18	4	15	18	4	15	20	4	13
SD OK/NOK	18	6	13	19	5	13	20	8	9
SD NOK/OK	18	5	14	19	5	13	19	4	14
ST OK/NOK	18	4	15	19	4	14	20	4	13
ST NOK/OK	18	4	15	19	4	14	20	4	13
90Deg OK/NOK - Tr1	18	4	15	19	4	14	20	4	13
90Deg OK/NOK - Tr2	20	4	13	21	4	12	23	4	10
SD OK/NOK - Tr1	20	4	13	19	4	14	19	4	14
SD OK/NOK - Tr2	20	4	13	20	4	13	23	4	10
ST OK/NOK - Tr1	19	5	13	20	7	10	20	8	9
ST OK/NOK - Tr2	19	5	13	20	7	10	21	7	9

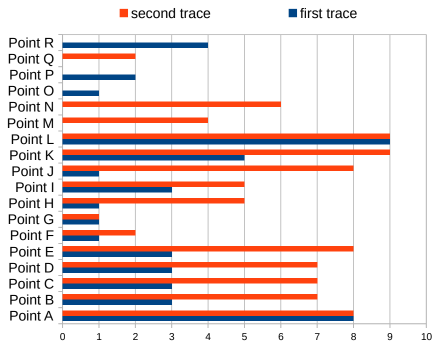
The survey carried out for this paper includes questions regarding which points experts would examine in more detail to determine if an individual trace is expected to produce an OK or not OK part. Figs. 5b, 5d and 5f show the POIs found when using the three methods described in Sect. 3.1 as distance measure between the average OK and average not OK time series for Alg. 4. Figs. 5a, 5c and 5e show which points experts wanted to examine more closely - based on looking at two different traces for which the marked timestamps as well as their drifts to the average OK and not OK time series were shown. The experts then had to choose for these two traces if they thought the trace



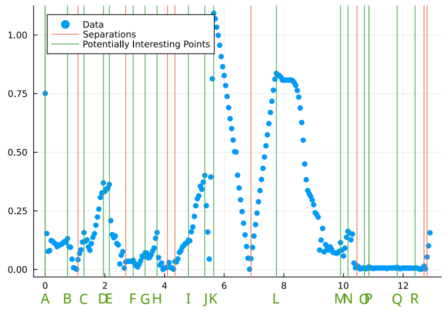
(a) Chosen Points From Fig. 5b



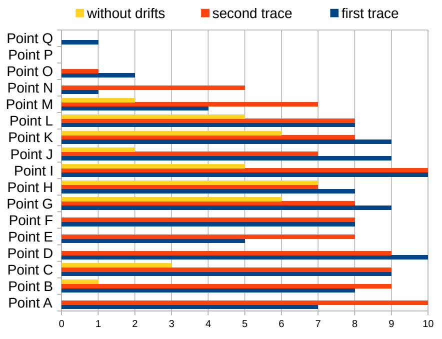
(b) POIs Using *90Deg* Method



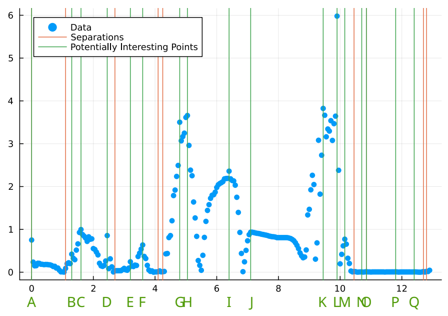
(c) Chosen Points From Fig. 5d



(d) POIs Using *SD* Method

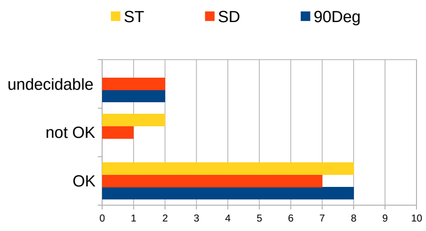


(e) Chosen Points From Fig. 5f

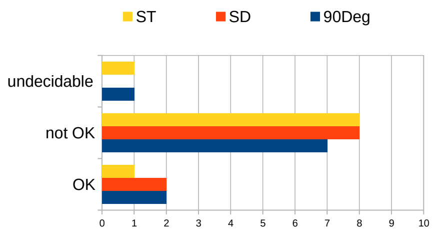


(f) POIs Using *ST* Method

represented a process run where the outcome would be OK or not OK. The results are shown in Fig. 6. As the first trace produces a correct part and the second trace produces a not OK part nearly all participants chose the correct option. Using only the points that have been selected by 5 or more participants in the survey (see Figs. 5a, 5c and 5e) and conducting the same analysis as for Tab. 1 rows 3–9 leads to the results shown in rows 10–15 - for each method the points selected for each trace are used ("Tr1"/"Tr2"). Overall, there is not much difference between the results achieved with the points selected by the experts in the survey and the automatically generated POIs when applied to all traces. Looking at the questions on how points that need to be analyzed in more depth are chosen reveals that experts choose the points based on (1) the length of the arrows/drifts, (2) where the difference between the arrows was sufficiently big, and/or (3) where arrows (drifts) go in different directions. Afterwards, the experts' prediction of the trace's class is based on which arrows are shorter (i.e., to which of the average time series chosen points are closer).



(a) Trace 1



(b) Trace 2

Fig. 6: Participants Prediction Results per Trace and Drift Visualization Method

5 Discussion

A limitation of the presented approach is that it can only be performed ex-post because detection of outlier traces (see Sect. 3.1), as well as calculation of average time series, depends on traces (or at least analyzed sensor data streams) being already completed. Additionally, the *90Deg* approach depends on data from timestamps to its left and right being available, which means it cannot be applied on the latest recorded point of the sensor data stream. Another limitation is that drifts are only visualized and analyzed for single sensor data streams. However, potentially multiple sensor data streams are recorded per trace which introduces the problem of determining which sensor data streams are important for detecting drifts for the whole process. An additional challenge in visualization is to consider the effects of large volumes of data (i.e., many traces being recorded). With the presented visualization approaches this can be tackled by utilizing grouping sequences of traces and adjusting how many traces are in a group together (therefore abstracting from information of the individual traces in favor of presenting properties of the groups) to allow for a comprehensible visualization.

The visualization approach presented in this paper shows drifts between different traces. This information is used to perform explainable outcome prediction based on the drifts at certain timestamps (the methods used to find these timestamps and determine the outcome of individual traces based on them is described in Sect.4.2). However, while the results for predicting the outcome are not satisfactory, as the quality of parts could only be predicted with $\approx 50\%$ accuracy, the approach nonetheless shows that it is feasible to work towards finding points and thus better prediction results in future work. Finally, we opted for a qualitative study to gather insights directly from end users which helped us in identifying strengths and weaknesses. But one potential problem of the questionnaire is, that it tests the understanding of the experts, and thus may lead to biased results. We will thus in future work improve the assessment of how the visualisation furthers data understanding, by conducting a quantitative study. In addition, we plan to improve the visualization by including multiple sensor data streams in the detection and visualization of drifts. Furthermore, we plan to extend the methods to make them applicable for runtime/live scenarios.

6 Conclusion

The paper proposes three approaches for the visualization of drifts in sensor data streams of different traces: “90 Degrees” (*90Deg*), “Shortest Distance” (*SD*), and “Same Timestamp” (*ST*). Furthermore, explainable outcome prediction is performed by utilizing how far traces drifted from OK / not OK traces.

The applicability of the visualization approaches was tested through implementation (contribution C1) and through a survey with 10 experts (partially contribution C3). 10 out of 10 experts considered *90Deg* and *SD* as better suited for identifying drifts than a raw trace visualization. Only 1 expert considered the raw visualization slightly useful in comparison to *ST*. However, the best method for visualizing drifts also depends strongly on the observed timestamp.

Contribution C2 of this paper, a way to use the visualization techniques at automatically derived points in the time series to predict the process outcome, was evaluated in the same survey (thus completing contribution C3). Here, the results of the survey show that using this approach allows for correctly predicting some of the trace outcomes ($\approx 48.65\%$–54.05%) and label some as undecidable ($\approx 10.81\%$–21.62%). But also a significant number of traces ($\approx 24.32\%$–40.54%) was assigned the incorrect outcome. However, even if the outcome prediction results are not satisfactory, the results show that when the right points are chosen, explainable predictions become feasible. In future work, we will thus focus on proposing methods for automatically finding points that yield better results for the outcome prediction.

References

1. Bose, R.J.C., van der Aalst, W.M., Žliobaitė, I., Pechenizkiy, M.: Handling concept drift in process mining. In: Advanced Information Systems Engineering. pp. 391–405. Springer (2011)
2. Figl, K.: Comprehension of procedural visual business process models. Bus Inf Syst Eng 59 p. 41–67 (2017)
3. Mallick, A., Hsieh, K., Arzani, B., Joshi, G.: Matchmaker: Data drift mitigation in machine learning for large-scale systems. In: Machine Learning and Systems. vol. 4, pp. 77–94 (2022)
4. Mehmood, T., Latif, S.: Dynamic big data drift visualization of cpu and memory resource usage in cloud computing. In: Artificial Intelligence Applications and Innovations. pp. 27–36 (2022)
5. O'Neill, M., Morgan, J., Burke, K.: Process visualization of manufacturing execution system (mes) data. In: SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation. pp. 659–664 (2021)
6. 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)
7. Pratt, K.B., Tschapek, G.: Visualizing concept drift. In: Knowledge Discovery and Data Mining. p. 735–740. KDD '03 (2003)
8. Sarnovský, M.: Concept drift visualization using feature importance on the streaming data. In: 2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI). pp. 000449–000454 (2022)
9. Sato, D.M.V., Barddal, J.P., Scalabrin, E.E.: Interactive process drift detection framework. In: Artificial Intelligence and Soft Computing. pp. 192–204 (2021)
10. Sharmin, M., Raij, A., Epstien, D., Nahum-Shani, I., Beck, J.G., Vhaduri, S., Preston, K., Kumar, S.: Visualization of time-series sensor data to inform the design of just-in-time adaptive stress interventions. In: Pervasive and Ubiquitous Computing. p. 505–516. UbiComp '15 (2015)
11. Stertz, F., Rinderle-Ma, S.: Detecting and identifying data drifts in process event streams based on process histories. In: Information Systems Engineering in Responsible Information Systems - CAiSE Forum. pp. 240–252 (2019)
12. Stertz, F., Rinderle-Ma, S., Mangler, J.: Analyzing process concept drifts based on sensor event streams during runtime. In: Business Process Management. pp. 202–219 (2020)
13. Taherdoost, H.: What is the best response scale for survey and questionnaire design; review of different lengths of rating scale / attitude scale / likert scale (06 2019)
14. Webb, G.I., Lee, L.K., Petitjean, F., Goethals, B.: Understanding concept drift (2017)
15. Yeshchenko, A., Ciccio, C.D., Mendling, J., Polyvyanyy, A.: Comprehensive process drift detection with visual analytics. CoRR **abs/1907.06386** (2019)