Assessing the Impact of Context Data on Process Outcomes During Runtime *

Matthias Ehrendorfer[1], Juergen Mangler[2], and Stefanie Rinderle-Ma[2]

[1] Faculty of Computer Science, University of Vienna, Austria
matthias.ehrendorfer@univie.ac.at
[2] Department of Informatics, Technical University of Munich, Garching, Germany
{juergen.mangler, stefanie.rinderle-ma}@tum.de

Abstract. The outcome of a process e.g., the quality of a produced part, constitutes a key performance indicator for process analysis and monitoring. Process outcomes are not only affected by process data, but also by data that is not associated with the process logic through decisions or task input. The rising temperature in a machine, for example, might cause deterioration of part quality. Assessing the impact of context data on the process outcome at runtime is particularly useful to reduce the reaction time to possible errors or deviations. However, as process models contain loops and decisions, grouping and making context data streams interpretable is not always straight-forward, especially under the condition that describing dependencies between context data and process data should be simple and flexible. The contribution of this paper is a classification of context data types, how they are connected to a process model, and how process models can be segmented into stages to group semantically related tasks. The impact of context data on the process outcome is then determined during runtime, i.e., as a process instance is progressing through these segments at runtime, impact calculations using context data can be gradually refined. The approach is prototypically implemented and applied to an artificial logistics and a real-world manufacturing data set.

Keywords: Manufacturing Intelligence · Runtime Process Analysis · Process Outcomes · Process Context Data · Impact Factors.

1 Introduction

Business processes are specified in the form of process models containing necessary tasks to reach a goal as well as the sequence of their execution. The process logic typically depends on data elements (e.g., the amount of a loan or the decision of process actors) that are available in the process. While tasks implementing a database access typically only receive data that can be utilised in

* 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 881843. This work has been supported by the Pilot Factory Industry 4.0, Seestadtstrasse 27, Vienna, Austria.

the process logic as a whole, tasks such as starting a machine and waiting for the machine to finish typically receive raw machine telemetry data that is discarded as it is not important for the process logic. Explicitly dealing with such telemetry data in the business logic is often not desirable (even to implement standardised data collection), as it complicates the process models and makes them much harder to maintain and improve. Another category of context data, is data that is never part of the process execution, but instead exists entirely outside of the scope of any process model. For example a hardware temperature sensor might continuously collect data while a machine is running, but the resulting data stream is never connected to a particular process instance.

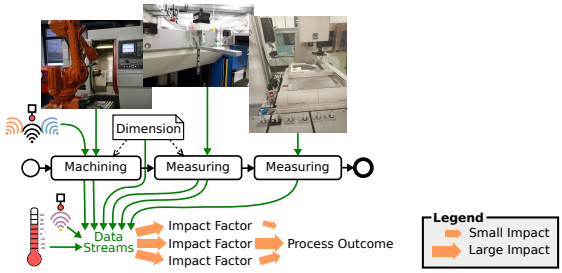

Fig. 1: Sample Process With Data Streams Collected During Process Run

Figure 1 shows a sample process from the manufacturing domain, which waits for (a) the machining of a part, (b) measurement results of a laser based optical micrometer, and finally (c) the tactile measurement results. While (a) from the perspective of the process is just about waiting until the task is finished, it yields gigabytes of data from the machining process itself and additional power and temperature measurement sensors. (b) on the other hand collects measurement information, that could be used for early termination of the process, but additionally gathers information about the temperature of the produced part. A part being too hot or too cold can have a serious impact on measurements, although this is not considered in the process. With (c) finished, a machined part as well as a detailed report about its quality is available. This is referred to as **process outcome**. Individual **data streams** (e.g., machining, power, temperature) are not part of the **data flow**, but nonetheless are important when reasoning why a certain outcome has been reached. Hence, **data streams** can be considered as **impact factors** for quantifying the process outcome.

Online (runtime) analysis of impact factors has the potential to predict outcomes, thus holding the possibility for optimising production processes regarding time and quality. Furthermore, analysing processes during their execution instead of ex-post enables to utilise information from unfinished process instances running in parallel. Another important aspect when dealing with impact factors is relevance. Not all impact factors might contribute equally to the quantification of the output. In previous work [2], first ex post analysis means for impact factors

of process outcomes based on annotating the process model is provided. However, methods for determining the importance of individual impact factors at runtime are missing. We tackle this research gap based on the following research questions:

- How can relevant impact factors be found in an online setting where process instances are only partly executed? How can we deal with decisions and loops?
- How does the completion of a trace including its outcome contribute to the confirmation or contradiction of the determined impact factors?
- How does the order in which traces are completed influence the certainty of the determined impact factors? How can this be used to reorder the traces to achieve a higher certainty in a faster way?

To tackle the above research questions, we introduce stages as a means to group tasks and their impact factors. Based on comparing information between stages of different instances we present static stage clustering and dynamic stage analysis approaches to predict the process outcome.

In order to evaluate the concepts presented in this paper, two data sets are analysed: (1) a synthetic simple logistic data set that comprehensibly demonstrates the main concepts, and (2) a real-world manufacturing data set with a multitude of sensors and high velocity machining data, that shows how complex multi-faceted data streams can be handled.

The remainder of the paper is structured as follows: Sect. 2 introduces fundamentals, Sect. 3 presents the approach, and Sect. 4 delves into how the clustering of impact factors can be realised, and how forecasts can be achieved. The approach is evaluated in Sect. 5 and the results are discussed in Sect. 6. Finally, related work is shown in Sect. 7 and the paper is concluded in Sect. 8.

2 Context Data Fundamentals

In general, impact factors are determined based on data that is available in the process. This data can stem from different data sources and ranges from data determining the control flow of the process to independent sensors measuring data streams that can influence the process. To handle these different types, **context data probes** are introduced to abstract from the underlying type of data when determining impact factors.

2.1 Context Data Probe Types

To track data in a process, different types of data probes can be distinguished (cf. Fig. 2):

(1) **Intrinsic Context Data Probes** (cmp.  in Fig. 2) describe data collected inside the process where an intrinsic motivation to obtain this data

exists stemming from the execution semantic of the process (i.e., a data element that is used to make a decision in the process or gives the termination condition for a loop). In literature this is often referred to as "process data" or "data elements".

(2) **Extrinsic Context Data Probes** (cmp. b in Fig. 2) describe data provided by tasks enacted in the process, but not manifesting in data elements of the process. Examples include tasks that interact with a machine or worklist where data is returned to the process.

(3) **Discrete Context Data Probes** are directly connected to the continuous stream of data from external sources not used in tasks of the process. Examples include data from temperature sensors or twitter feeds which might influence the execution of the process. Two different types exist:

- **Instance Based Discrete Context Data Probes** (cmp. c in Fig. 2) track the continuous data stream during the whole execution time of the instance. This allows for the collection of data streams from continuous data streams not connected to any of the tasks in particular but possibly being able to influence the process instance during its runtime.
- **Task Based Discrete Context Data Probes** (cmp. d in Fig. 2) only track the continuous data stream during the execution of a specific task. This enables collecting parts of data streams from autonomous sources that only have an influence when certain operations are performed.

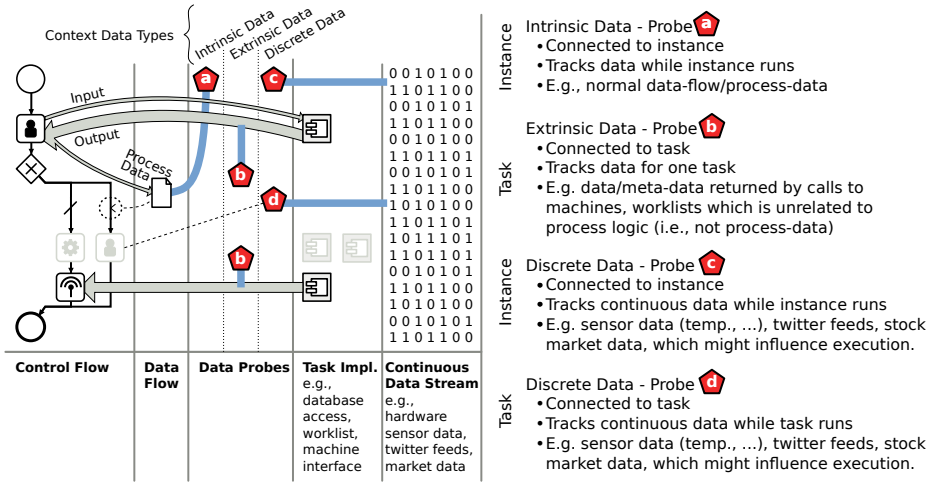

Fig. 2: Types of Data in the Process Context

2.2 Impact Factors and Impact Profiles

This section explains impact factors and profiles as introduced in [2] and depicted in Fig. 3. Data probes produce homogeneous data streams, which are

then aggregated. This can happen either with simple (avg, median) or complex domain specific aggregation functions depending on the use case, similarly to calculating key performance indicators, cf. [7]. An impact factor itself can be an aggregation, e.g., inside a machine the temperature might be taken at various locations to account for local heat build-up. The impact factor combines the data from all temperature sensors. Finally, different impact factors are weighted and combined to form profiles. Profiles can either exist for individual tasks or at the instance level.

How to derive the weights between impact factors is one of the contributions of this paper, and will be explained in detail in the next chapters. It is assumed that there is a notion of good or bad outcome: i.e., in a manufacturing process, after quality control it is known if a part is good or bad. We can thus summarise that the following domain specific input to derive impact profiles is necessary:

- A superset/list of data streams which might potentially influence outcome.
- A function how to aggregate each homogeneous data stream.
- A function how to aggregate one or more data impact values (even if the values e.g., derive from different sensor types).
- A set of impact factors that contribute to an impact profile.
- A binary notion of process outcome: good/bad.

The weights for the impact profile function are then calculated in a way so that good parts yield a result that tends towards 1 and bad parts 0.

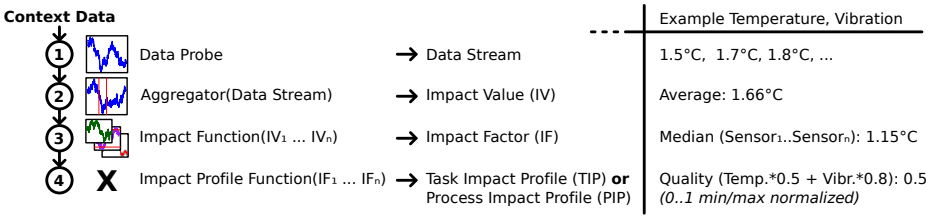

Fig. 3: Impact Profiles and Related Concepts

3 Runtime Context Data Analysis

The fundamentals of context data as used in this paper are explained in Figs. 2 and 3: (a) which data types can occur in the process context and (b) how to handle data streams that are collected during process execution. Figure 4 shows a concrete example of a process in the manufacturing domain where external data is collected in some tasks. Individual data streams can then be aggregated and combined as outlined in Fig. 3 and performed in the example in Fig. 4 where different ways of building impact factors from data streams are shown. The impact factors are then used in further steps of the approach.

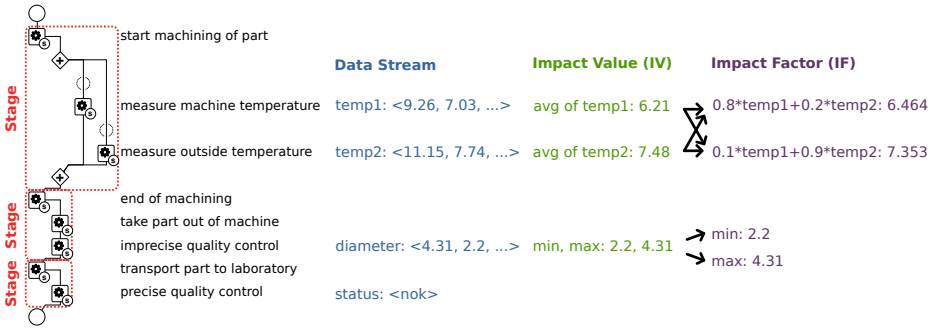

Fig. 4: Running Example Process

3.1 Comparing Process Instances - Stages

During runtime, a multitude of process instances might be active and in different states of their execution. An execution state is defined by the set of tasks that are currently executed. As the definition of impact factors depends on the tasks, the different execution states result in a varying number of impact factors for the currently active process instances. This can aggravate the comparison of the impact factors over a set of process instances. Hence, we suggest the usage of *stages* that reflect certain execution states in a process and enable to cluster the running instances along these states. Figure 4 depicts the running example process with three stages reflected by boxes.

Stages are especially important when process models allow different behaviour for individual instances. For example, a manufacturing process might skip steps or run through certain steps in a loop, e.g., for iterative refinement of certain aspects that require constant adaptation of manufacturing parameters. Obviously, only process instances being in the same stage can be compared as different control flow behaviour can affect the collected data. However, even with different process models (e.g., different versions of a process), similar process instances might be comparable if they share certain stages.

Stages are user defined at the process model level, and consist of one or several tasks, based on semantic affiliation of included data (e.g., same source, collected in same step)[3]. If, for example, one overall machining operation consists of multiple tasks which represent different machining programs applied on a single piece of raw material, and supervised by a set temperature and vibration sensors, they can be grouped in a stage by the process designer. At the process instance level, a stage is complete, when all tasks contained in the stage have been completed. This constitutes a trigger point for (a) forecasting the next stage, and (b) refining the forecasting data set for the finished stage (see Sect. 4.2).

[3] In future work, we aim at the automatic definition of stages based on process abstractions [9] or inspired by automatic approaches such as [6].

Predicting how an upcoming stage might contribute to the outcome depends on one or several stages that have been already finished. When analysing the data modified in finished stages, two types of data can be identified.
Static Stage Clustering: If data points in a set of stages are similar, they can be grouped. Future stages of instances being in the same groups might also contribute to outcome similarly. Therefore instances with such static stages are clustered (see Sect. 4.1 on how a data stream is analysed to cluster instances).
Dynamic Stage Analysis: If a process instance has new data points compared to instances that are in an earlier stage, the difference constitutes a potential progression an early-stage instance might take. An outcome prediction based on this potential progression is possible when comparing instances which are in different stages (see Sect. 4.2).

4 Realisation

Two techniques are employed to realise the introduced concepts, i.e., clustering and refinement of the importance of impact factors when a process instance progresses from one stage to the next one.

4.1 Clustering

Data streams need to be grouped to find out which ones are important for the outcome of a process instance. Without results from earlier process executions it is necessary to identify the streams being similar for "normal" process executions and others deviating from the norm. During clustering, points being close to each other based on a distance metric are grouped. This grouping is utilised by assuming that data streams that can easily be clustered are more important for the outcome. Therefore, the following steps are performed:

```
process_instance = ((DS,A)+,IFU)*) # each process instance contains all
    impact factor (IF) definitions consisting of data streams (DS),
    aggregations (A), and impact functions (IFU)
possible_DS_combinations = all possible combinations of data streams

for DS_set in possible_DS_combinations
  all_IF_lists=[]
  for process_instance in all_process_instances
    IF_list=[]
    for ((DS,A)+,IFU) in process_instance
      if((DS,A)+ contains_all_DS_of DS_set)
        IV_list=[]
        for (DS,A) in (DS,A)+
          IV = aggregate(DS,A)
          IV_list.push(IV)
        IF = create_impact_factor(IV_list,IFU)
        IF_list.push(IF)
    all_IF_lists.push(IF_list)
  params = determine_clustering_params(all_IF_lists) # kNN plot
  cluster_assignment = build_clusters(all_IF_lists,params) # assign
      cluster to each process_instance using DBSCAN
  for cluster in cluster_assignment
    cluster_quality(all_IF_lists,cluster) # silhouette value
add list of assigned clusters & their quality  to each process_instance
```

Algorithm 1.1: Static Stage Clustering

Looking at the running example (Fig. 4), two temperature based impact factors and two impact factors based on the diamater are available. Therefore, a process instance is assigned to two clusters (one for the temperature data streams and one for the diameter data stream). The concrete clustering technique is not important for the general idea. However, as different techniques require different information to perform them, two techniques are considered for this paper:

- The k-means algorithm, following the argumentation in [3], is a well explored approach. On the flip side, it requires the number of clusters (and their initial centre points) as input. Using this clustering technique for finding similar data streams is therefore difficult as it is not known beforehand how many clusters should be found as stream data can show a multitude of different behaviours. Even with methods existing for determining the number of clusters, this technique is not suitable for the intended purpose.
- The DBSCAN algorithm [8] finds clusters based on the distance between data points. These distances are used to determine which points form a cluster and which are too far apart. Therefore, it is not necessary to provide the number of clusters as input. However, the *epsilon* value needs to be provided which defines the neighbourhood of points used for finding points being in the same cluster. This value can be found using a k Nearest Neighbours (kNN) graph if no value from expert knowledge is available.

Based on these considerations we opt for the DBSCAN algorithm. Concerning a quality measure for the whole clustering as well as for individual clusters, the silhouette value is used. The silhouette value can be calculated for each data point and is between -1 and 1. Low values are obtained if points from other clusters are closer than the ones of the same cluster and high values are gained if the point is close to points from its own cluster. Therefore, the silhouette value of a cluster or of all points gives an idea of how close data points are to other points in the same cluster (i.e., how well clustering works).

4.2 Stage Progression

All steps described before are performed in one go where some data streams are already available while other information is not. The final step of the approach presented in this paper is to have individual instances progress in their execution. Two cases exist: for a stage where some instances already have impact factors / data clusters, a forecast for the outcome of the stage can be derived. For stages, where this is not the case, forecast is not possible. The information about available clusters is used (as described in Alg. 1.2) to determine the overall score of a process instance (representing the impact profile) taking into account the importance of different impact factors and their values for the specific instance.

```
process_instance = ((DS,A)+,IFU)*, cluster_assignment) #
    process_instances additionally contain their cluster assignment
possible_DS_combinations = all possible combinations of data streams

for DS_set in possible_DS_combinations
  for cluster in clusters
    stat_value = get_static_value(cluster)
    if(cluster in cluster_assignment)
      dyn_value = get_dynamic_value(cluster)
      update_overall_score(stat_value,dyn_value)
```

Algorithm 1.2: Dynamic Stage Analysis

As also described in Sect. 4, the static value of stages is obtained by using the silhouette value of the corresponding cluster based on the group of unfinished process instances. The dynamic value, is based on already finished process instances. Therefore, the share of positive outcomes of the corresponding clusters represents the dynamic value and is combined with the static value to determine the value added to the overall score for determining the outcome of the examined process instance.

5 Evaluation

5.1 Settings

One evaluation scenario is a manufacturing process where a part is produced by a machine tool and afterwards measured twice. Data about the manufacturing process is therefore collected (1) during the manufacturing of a part, (2) during the fast, but imprecise measurement directly after the manufacturing of a part, and (3) during the slow, but precise measurement of the part performed independent of the manufacturing of a part. Parts being taken out of the machine can have a metal chip from the machining on it requiring special handling. After the production step, process instances of the manufacturing process are in a stage where all data (i.e., machining and the fast, but imprecise measurement data) is already collected, but the outcome (i.e., chip occurrence or quality control test result) is still unknown. The data streams used for the evaluation are the workload of the drive (aaLoad) and the axis speed (aaVactB) for the X, Y, and Z axis together with the actual speed of the spindle (actSpeed) and the workload of the spindle (driveLoad) from the machining of the part and the measurement values from the fast, but imprecise measurement which measures the silhouette of the part. For all of these values the minimum, maximum, average, and weighted average (which tries to tackle irregular machine tool measurements) are used to get characteristic values of the timeseries for clustering. Furthermore, the weighted average of an important segment of the fast, but imprecise measurement is used for determining the outcome of a quality control test.

Another data set used for the evaluation is adapted from a realistic container transportation case described in [1]. The process includes the loading of a vehicle which afterwards moves towards its destination. During this journey, the temperature is constantly measured. When the temperature is beyond a certain

point for a certain period of time, the vehicle has to return to its origin. Otherwise it continues towards the destination where the container is unloaded. As this process only contains one data stream that is measured (i.e., the temperature) it was decided to additionally use the temperature of each third of the measurement interval as an individual data probe (resulting in 4 data probes) to showcase the approach. Again, minimum, maximum, and average are chosen for obtaining values for clustering the time series. The outcome of the process is defined by normal cases and exceptional cases (i.e., cases where the vehicle has to return to its origin). The process instances are in a stage where the temperature is already measured. However, it is not known if the vehicle has to return to its origin (negative case) or if it is able to stay on the route to its final destination (positive case).

5.2 Evaluation Process

As described in Sects. 3 and 4, the first step of assessing the impact factors of data streams on process outcomes during runtime is to obtain the static characteristics by clustering traces based on the available data. This is done individually for each data stream meaning process instances are clustered multiple times (i.e., once per data stream). The DBSCAN clustering algorithm is used because the number of clusters is not previously known. The *epsilon* value (needed for performing DBSCAN) is determined using kNN plots and finding the elbow in the graph.

Clustering provides a silhouette score describing how close data points in one cluster are together compared to other clusters. The silhouette value can be given for the overall clustering result of a data stream as well as for individual clusters. As explained in Sects. 3 and 4, clusters sticking closer together are assumed to also be more important impact factors for the outcome of a process.

The first example from the manufacturing scenario is shown in Fig. 5. Here, the outcome is represented by the occurrence of a chip on the part. Figure 5a shows the development of the importance of data streams for the outcome after the specified number of process instances have been continued (therefore considering the static characteristics as well as the change of the dynamic characteristics of different data streams). Obviously, the imprecise measurement is the most important impact factor for this outcome. The development of the overall score of process instances based on static and dynamic characteristics is shown in Fig. 5b. The scores of process instances with positive (i.e., no chip on the part) and negative (i.e., chip on the part) outcomes differ from a certain point on. This is depicted by green (positive outcome) and red (negative outcome) boxplots for different numbers of finished process instances. Lower scores signal a negative outcome while higher scores signal a positive outcome.

However, it is also important to get to a point where cases can be distinguished as fast as possible (i.e., by having to finish as few process instances as possible). The approach presented in this paper selects the next process instance up for continuing execution based on the clusters to which the data streams are assigned by choosing the one having the overall highest impact. In contrast to this strategy, continuing the execution of process instances randomly (Fig. 5c)

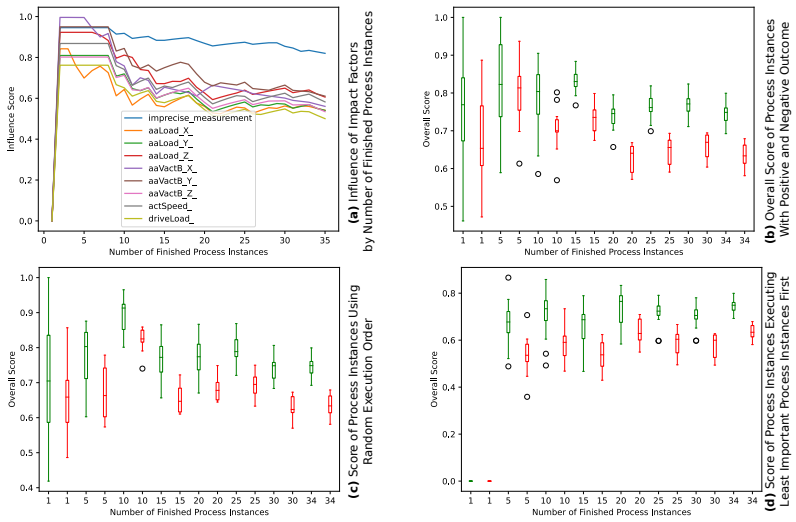

Fig. 5: Chip Occurrence in Parts of Batch 15

or always choosing the one with the lowest overall impact (Fig. 5d) leads to different behaviour. It can be seen that process instances with positive outcomes can be distinguished from ones with negative ones at an earlier point in time (approximately after 15 process instances have been finished) when choosing process instances with high impact for execution as shown in Fig. 5b in contrast to different ordering techniques as shown in Figs. 5c and 5d. Additionally, the order of process instances can be chosen before any process instances are starting to continue or it can be adapted each time another process instance finishes and therefore more information is available. However, this difference is not discussed due to shortage of space. If not specifically described otherwise all following figures show the approach when the most impactful process instance is chosen and continued after the previous process instance has finished.

Using the same data set as above, but another outcome (i.e., the passing of a specific quality control test) leads to the results shown in Fig. 6. For batch 15 positive and negative outcomes are not clearly distinguishable (see Fig. 6b) and no data stream clearly important for the outcome can be found (see Fig. 6a). However, for batch 14, the overall score of individual process instances can be used to distinguish between cases with different outcomes (see Fig. 6d). Furthermore, it can be seen in Fig. 6c that even if no single important data stream can be identified, there is a group of data streams (actSpeed, aaLoad_Z, and aaVactB_X) being more important than the other ones.

Using the logistics data set for the evaluation leads to the results shown in Fig. 7. Figure 7a shows the development of the impact factors of the data streams that are chosen for the logistics data set as given in the scenario description. Furthermore, Fig. 7b shows that the overall score of process instances with a positive outcome (i.e., normal cases) achieve higher values than ones with

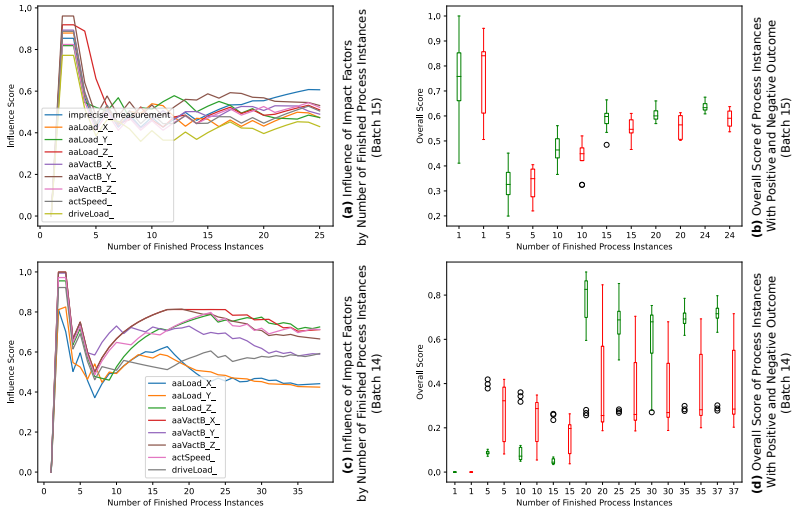

Fig. 6: Quality Control Test Result in Parts of Batch 14 and 15

a negative outcome (i.e., exceptions) after the initial information gained from clustering is refined by executing additional traces (i.e., about half the process instances have been executed). However, as with the last example, no single data stream can be highlighted as most important.

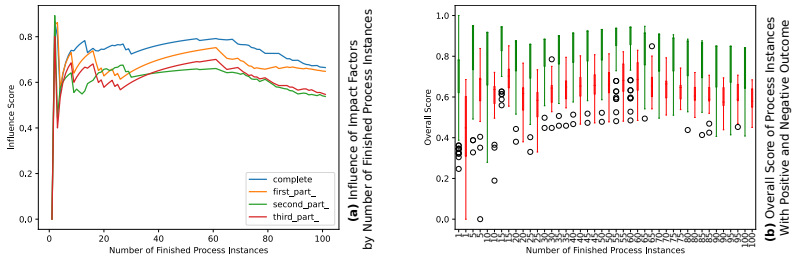

Fig. 7: Completion of Route for Logistics Use Case

Overall, the evaluation shows, that it is possible to identify the importance of different impact factors for outcomes of a process at runtime. Using the identified influence of the impact factors on the outcomes allows to calculate an overall score. The approach is evaluated using different domains and shows its applicability by making it possible to distinguish between process instances with a positive outcome and ones with a negative outcome after the initial importance of impact factors has been refined by finishing some initially unfinished process instances. The evaluation also shows that the order in which traces are finished has an effect on how early different outcomes can be identified.

Code and data used for the evaluation along with instructions how to use it is available on gitlab [4]. The manufacturing data is based on the process logs available at cpee.org [5] [6]. The logistics data is based on the case described in [1].

6 Discussion

The evaluation shows that impact factors along with their influence on the outcome can be found. However, supposing that the order in which process instances are continued can be freely defined, the question emerges how the determination of impact factors can be sped up. A possibility is to reorder process instances such that always the one for which the data streams are assigned to the most promising clusters is continued next. As shown in the evaluation this allows to faster distinguish between process instances having a positive/negative outcome. The order can be set either before executing any process instances or it can dynamically change each time new information is available (i.e., when a process instance is continued). This also has implications for real-world applications. In the manufacturing domain it might be necessary to know the order in which parts should be measured beforehand. For static processes this cannot be adapted. However, more dynamic processes which allow to adapt processes based on new insights may support changing the order during the process. An example for a static scenario where the order in which process instances are executed has to be known beforehand would be a robot taking parts from a conveyor belt in the order in which they have been placed. In contrast to this, a robot which picks parts from a tray based on the information available in the process only needs the information which part to pick right before picking.

Concerning the data set of the manufacturing process, two batches are used for the evaluation. One is used to evaluate the described approach for finding impact factors and their importance for two different outcomes (i.e., occurrence of a chip and passing of a quality control test). The other one is used to perform the evaluation for passing a quality control test with different data and the results are compared to each other for validation. The logistics data set is used to show that the approach is applicable to multiple areas where data inside a process is measured over a time period. Another area matching this description is the medical domain where process instances correspond to the treatment of one person and different data such as the temperature or the blood pressure of the patient is measured multiple times. Other domains where the impact of different data streams on the outcome should be determined could also be suitable.

As discussed, the presented approach has certain limitations regarding the scenario. To use the knowledge gained from process instances being slightly ahead of others it has to be possible to intervene in the latter ones. This allows to use information gained from further advanced process instances to adapt

[4] `https://gitlab.com/me33551/runtime_impact_factor_assessment` [Online; accessed 12-Aug-2021]

[5] `https://cpee.org/~demo/DaSH/batch14.zip` [Online; accessed 12-Aug-2021]

[6] `https://cpee.org/~demo/DaSH/batch15.zip` [Online; accessed 12-Aug-2021]

process instances which are similar to improve the outcome or at least be prepared for formerly unexpected events. However, this does not necessarily mean that process instances influence each other, it is just about identifying similar instances to improve prediction of the outcome. Regarding the complexity of the proposed algorithms, Alg. 1.1 analyses each instance for every combination of data streams which may lead to long execution times for big data sets with many data streams. Algorithm 1.2 is also depending on the number of possible data stream combinations (but only once because instances progress individually).

Future work will deal with how stages are best defined and if there is a way to identify them automatically instead of manually. Furthermore, the composition of impact factors based on data probes needs to be developed towards the direction of finding meaningful combinations instead of needing domain knowledge.

7 Related Work

Recently, process mining and predictive process monitoring approaches have started to consider and analyze process perspectives beyond control flow, including process data [5]. Also external data such as time series data is exploited for detecting concept drifts during runtime [10]. In contrast to these approaches, this paper tries to determine how much impact data streams collected during the process have on the outcome. The survey presented in [11] compares different outcome-oriented predictive process monitoring techniques. However, existing approaches do not consider the impact of continuous data streams from external data sources on the outcome of the process. Anomaly detection for manufacturing systems based on sensor data is, for example, tackled by [4]. However, the process aspect and particularly the impact of the sensor streams on the process outcome are not considered. [7] defines an ontology for process performance indicators (PPIs), together with templates and patterns. The PPIs can be defined to aggregate observations in the process. This constitutes valuable input for aggregating impact factors after being transferred to work on external data streams. [6] presents an approach to find stages in a process by automatically maximising the measure of modularity which describes a high density of connections within a stage and a low number of edges between stages. However, [6] only considers the control flow of processes. Therefore, external data which is important for the definition of stages, is not taken into account. The definition of stages is also connected to process abstractions. A survey on process abstractions is provided in [9], also discussing why, when, and how abstraction is applied. For this paper, abstraction supports the focus on the data perspective. The abstraction is done by identifying tasks containing data streams applying to the same abstract steps of the process and group them together in one stage.

8 Conclusion

Knowing the outcomes of process instances while they are still executed bears advantages for process operators. This paper presents an approach to assess the

impact of data streams on process outcomes during runtime. Clustering individual data streams allows to determine the initial importance of different impact factors i.e., their share in influencing the outcomes. This is initially only based on the available data from unfinished process instances. Afterwards, process instances being continued are used to refine the initial assessment. Furthermore, it is shown that when the reordering of traces is possible, it is beneficial to finish process instances where the data streams belong to clusters that are promising candidates for important impact factors.

To answer the research questions three concepts are presented in this paper. Firstly, in order to reduce the complexity of a high number of process instances being executed until a certain task, stages are used to support the comparison between different instances that are comparable regarding the collected data. Secondly, static characteristics of impact factors for the process outcome are used to describe their maximum impact on the outcome. Thirdly, dynamic characteristics are used to deduce the actual impact of different factors on the outcome. In contrast to static characteristics which are determined only with unfinished process instances and stay the same, dynamic characteristics are adapted based on the actual outcomes of process instances finished over time.

The approach presented in this paper is evaluated using two batches of a real-world data set from the manufacturing domain including multiple data streams as well as one data set from the logistics domain to show the applicability of the approach for other domains where continuous data streams are included.

References

1. 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: Advanced Information Systems Engineering. pp. 68–84 (2014)
2. Ehrendorfer, M., Mangler, J., Rinderle-Ma, S.: Sensor Data Stream Selection and Aggregation for the Ex Post Discovery of Impact Factors on Process Outcomes. In: CAiSE Forum. pp. 29–37 (2021)
3. Faber, V.: Clustering and the continuous K-means algorithm. Los Alamos Science **22** (Jan 1994)
4. Kammerer, K., Hoppenstedt, B., Pryss, R., Stökler, S., Allgaier, J., Reichert, M.: Anomaly Detections for Manufacturing Systems Based on Sensor Data—Insights into Two Challenging Real-World Production Settings. Sensors **19**(24), 5370 (2019)
5. Mannhardt, F.: Multi-perspective Process Mining. Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven (Feb 2018)
6. Nguyen, H., Dumas, M., ter Hofstede, A.H.M., La Rosa, M., Maggi, F.M.: Mining Business Process Stages from Event Logs. In: Advanced Information Systems Engineering. pp. 577–594 (2017)
7. del Río-Ortega, A., Resinas Arias de Reyna, M., Durán Toro, A., Ruiz-Cortés, A.: Defining Process Performance Indicators by Using Templates and Patterns. In: Business Process Management. pp. 223–228 (2012)
8. Schubert, E., Sander, J., Ester, M., Kriegel, H.P., Xu, X.: DBSCAN Revisited, Revisited. ACM Transactions on Database Systems **42**(3), 1–21 (2017)

9. Smirnov, S., Reijers, H.A., Weske, M., Nugteren, T.: Business process model abstraction: a definition, catalog, and survey. Distributed and Parallel Databases **30**(1), 63–99 (Feb 2012)
10. 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)
11. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. ACM Trans. Knowl. Discov. Data **13**(2) (Mar 2019)