



Integrated detection and localization of concept drifts in process mining with batch and stream trace clustering support

Rafael Gaspar de Sousa ^{a,*}, Antonio Carlos Meira Neto ^a, Marcelo Fantinato ^{a,*},
Sarajane Marques Peres ^{a,*}, Hajo Alexander Reijers ^b

^a School of Arts, Science and Humanities, University of São Paulo, Brazil

^b Department of Information and Computing Science, Utrecht University, The Netherlands

ARTICLE INFO

MSC:

68U35

68T05

Keywords:

Concept drift

Trace clustering

Process mining

Business processes

Data mining

ABSTRACT

Process mining can help organizations by extracting knowledge from event logs. However, process mining techniques often assume business processes are stationary, while actual business processes are constantly subject to change because of the complexity of organizations and their external environment. Thus, addressing process changes over time – known as *concept drifts* – allows for a better understanding of process behavior and can provide a competitive edge for organizations, especially in an online data stream scenario. Current approaches to handling process concept drift focus primarily on detecting and locating concept drifts, often through an integrated, albeit offline, approach. However, part of these integrated approaches rely on complex data structures related to tree-based process models, usually discovered through algorithms whose results are influenced by specific heuristic rules. Moreover, most of the proposed approaches have not been tested on public true concept drift-labeled event logs commonly used as benchmark, making comparative analysis difficult. In this article, we propose an online approach to detect and localize concept drifts in an integrated way using batch and stream trace clustering support. In our approach, cluster models provide input information for both concept drift detection and localization methods. Each cluster abstracts a behavior profile underlying the process and reveals descriptive information about the discovered concept drifts. Experiments with benchmark synthetic event logs with different control-flow changes, as well as with real-world event logs, showed that our approach, when relying on the same clustering model, is competitive in relation to baselines concept drift detection method. In addition, the experiment showed our approach is able to correctly locate the concept drifts detected and allows the analysis of such concept drifts through different process behavior profiles.

1. Introduction

In today's organizations, business processes are often changed in response to new scenarios arising from, e.g., customer needs, resources and personnel involved, economics, technology, or legislation [1]. The current global health crisis is a clear example of a new scenario that has led to quick adaptations in many processes. These new scenarios influence the dynamics of organizational operations and hence lead to changes in the behavior of the underlying processes, which are called process *concept drifts*. In fact, the success of an organization increasingly relies on its ability to react and adapt to new scenarios [2]. Thus, the capability to quickly detect and understand process concept drifts has become a competitive edge for organizations. By being able to detect and

* Corresponding authors.

E-mail addresses: rafaelgaspar@hotmail.com.br (R.G. de Sousa), m.fantinato@usp.br (M. Fantinato), sarajane@usp.br (S.M. Peres).

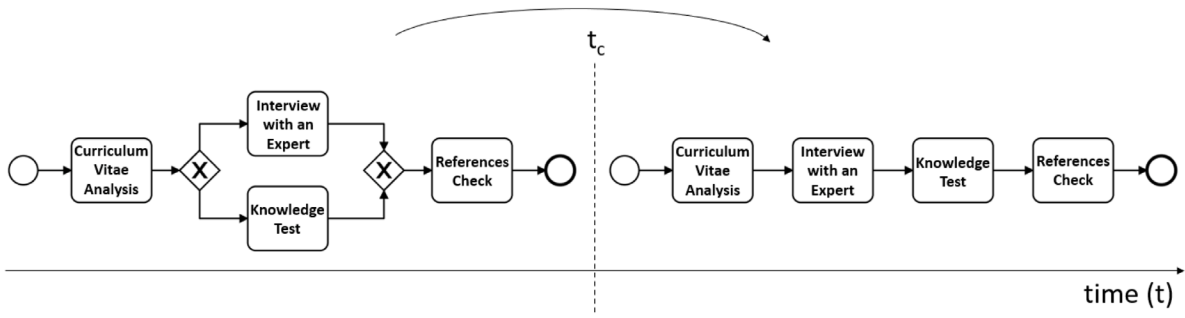


Fig. 1. Illustrative example of concept drift in a process.

understand process changes, organizations can quickly react to correct, adapt, or improve their processes whenever needed. Concept drift treatment is expected to be manageable mainly in online environments, which present even greater challenges [3]. In these environments, the challenges involve data volume, velocity, volatility, and uncertainty in behavior persistence [4,5]. Efficiently processing large amounts of data within limited timeframes is essential due to volume and velocity. Volatility indicates diminishing utility of past data over time, resulting in limited information for analysis. Additionally, concept drift requires persistence. In organizational environments, sporadic yet acceptable behaviors may arise, posing complexities in determining the appropriate temporal granularity for addressing concept drift.

Process mining has been supporting organizations to handle their business processes in descriptive, predictive, and prescriptive ways. As summarized by de Sousa et al. [6], following van der Aalst [7,8]’s definitions, process mining relies primarily on the concepts of event, case, trace, log, and attribute. An *event* is the occurrence of a business process activity at a given time, performed by a given resource, at a given cost. A *case* corresponds to a process instance and comprises events such that each event relates exactly to a case. A *trace* is a mandatory *attribute* of a case and corresponds to a finite sequence of events such that each event appears only once. An *event log* is a set of cases such that each event appears only once in the entire event log. Each event in the event log comprises a set of non-mandatory *attributes* such as identifier, timestamp, activity, resource, and cost. Cases can also have non-mandatory *attributes*, often related to domain-specific data.

Although the use of process mining to support organizations is increasing [7,9], dealing with process behavior changes is still a challenge. In fact, the vast majority of process mining methods apply exclusively to stationary environments where the underlying concepts do not change over time [10–12]. However, event log data, recorded by information systems and used as raw material for process mining, is indeed subject to concept drift. In process mining, concept drift refers to changes over time in control-flow logic, resource allocation, case performance dynamics, or other domain-specific non-stationary phenomena.

Based on Bose et al. [13], we argue the key challenges for concept drift in process mining center on four tasks: (i) *detection* of when the process has changed; (ii) *localization* of the elements involved in the process change; (iii) *characterization* of the type of process change; and (iv) *explanation* of the reason for changing the process. To illustrate these tasks, Fig. 1 shows two models of a process that experienced a concept drift. Previously (cf. model on the left), an XOR gateway shows that either activity *Interview with an Expert* or activity *Knowledge Test* needed to be performed per process instance; later, after instant t_c (cf. model on the right), all activities started to be performed sequentially. In this example, the concept drift detection should identify the process changed at instant t_c ; the concept drift localization should identify the change involves activities B and C; the concept drift characterization should identify the change refers to the sequencing of activities that were previously exclusive, i.e., a change in the control-flow perspective took place; and, finally, the concept drift explanation should present the reason for the change, which could be, hypothetically, the effect of an amendment due to related legislation.

Concept drift detection is the most addressed task in the studies found in the process mining literature. About half of the proposed approaches focus exclusively on concept drift detection. However, pure concept drift detection may not add as much value to organizations [14–16]. The second most addressed task is concept drift localization, which is often solved based on information previously extracted for concept drift detection, characterizing an integrated approach. However, some integrated approaches rely on complex data structures related to tree-based process models, usually discovered through algorithms whose results are influenced by specific heuristic rules. Furthermore, most of these integrated approaches are not designed to run online. Finally, comparative analysis of results among proposed approaches is difficult as most of them have not been tested on public true concept drift-labeled event logs commonly used as benchmark.

In this article, we propose an approach to the integrated detection and localization of concept drifts in process mining, which can be supported by both batch and stream trace clustering.¹ To achieve this, our approach is based exclusively on trace clustering [17]. Trace clustering is used to find patterns and split event logs into groups representing similar process behavior [18]. Our approach implements trace clustering by mapping traces into a feature vector space based on a specific process profile (e.g., activities, resources, transitions, etc.) before submitting them to a vector similarity analysis method.

¹ Code available at https://github.com/pm-usp/concept_drift/tree/main/trace_clustering.

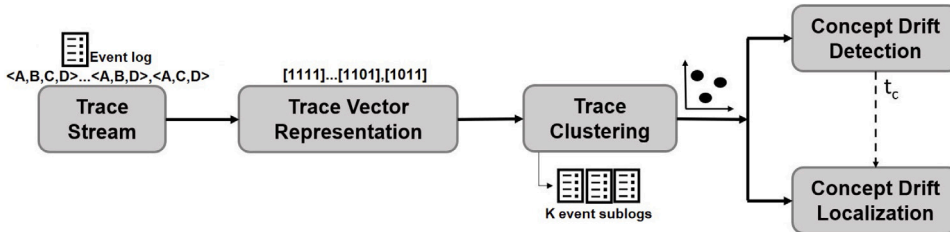


Fig. 2. Pipeline of the proposed approach.

In event logs with high trace variability, the process underlying the event log may be complex and poorly structured [19]. In such cases, trace clustering can isolate different process behaviors into groups [17,20–23]. Concept drift detection and localization benefit from trace clustering, as concept drifts may occur only in some of such behaviors. Process model analysis also becomes easier by allowing business analysts to analyze the specific contexts in which concept drifts occur.

As illustrated in Fig. 2, the approach we propose allows for an efficient resolution of two interrelated tasks regarding concept drift: *concept drift detection* and *concept drift localization*. Both tasks work upon information from trace clusters representing process behavior. For this, the incoming trace is mapped into a *trace vector representation* taken as input by the *trace clustering* method. Our approach leverages the clusters' centroids at a given time to solve both tasks. First, concept drift detection relies on data on the variation of centroid positions over time to verify whether the current process behavior observed in the event log reveals significant changes in relation to the behavior observed previously. Second, concept drift localization compares the differences in centroid components to identify the process elements involved in behavior changes (considering before and after) related to the detected concept drift. Assuming the incoming traces as an online *trace stream*, the trace clustering method is able to properly consider the influence of past data on current data. Event sublogs resulting from splitting traces into clusters can be used as simplified input for process model discovery and visual analysis in concept drift context.

The approach presented herein primarily addresses two key issues in improving concept drift solutions in process mining: (a) the need for an efficient and integrated method to tackle multiple facets of the problem, and (b) effectively handling the high variability of traces in the analyzed process, including the challenge posed by rare traces. To achieve this, the approach involves running a single strategy in processing event log data, using a clustering algorithm. The result is a quantizer data model that enables the identification and localization of concept drifts. Moreover, this model partitions the event log and effectively mitigates data space variability, thereby allowing the identification of salient but local behaviors through the analysis of quantized information. Nevertheless, the cluster-based approach needs the establishment of two assumptions: (a) prior knowledge regarding the activities associated with the underlying business process recorded in the event log, and (b) the analysis being conducted based on a complete trace within the trace stream model. Although these assumptions restrict the applicability of the approach to scenarios such as critical process monitoring, they are deemed reasonable for organizations with well-developed business processes, which are typically influenced by intricate phenomena including information system releases [15], technological advancements, regulatory and managerial alterations [24–26], as well as market demands.

Our approach extends our previous work [6], which applied the k-means++ batch clustering method using a fixed data windowing strategy to handle concept drift. As new contributions, we (i) applied the CluStream stream clustering method to handle concept drift through an inherently online method, and compared the results obtained with window-based k-means++ and with a non-clustering version of our approach, and (ii) expanded the experiments, to increase the reliability and robustness of our results, through synthetic event logs with higher trace variability, as well as through real-world event logs.

The rest of this article is organized as follows. Section 2 discusses related work on concept drift in process mining. Section 3 summarizes our approach and details the trace vector representations used, the clustering strategies applied to batch and stream trace clustering, and the proposed methods for both concept drift detection and localization. Sections 4 and 5 report the experiments on synthetic and real-world event logs, respectively. Finally, Section 6 concludes this article.

2. Related work

Approaches to dealing with concept drift in process mining have been proposed at least since 2011. Concept drift *detection* has been the most addressed task, followed by concept drift *localization*. Both concept drift detection and localization are usually resolved in an integrated way. Concept drift *characterization* began to be addressed in 2018, while concept drift *explanation* was first addressed only in a recent study (in 2021). To organize a picture of concept drift efforts in process mining, we classified the related works considering three categories of concept drift detection methods (statistical testing-based, similarity-based, and time series-based) and two detection processing ways (offline, and online with event stream or trace stream).² Tables 1 and 2 summarize the 29 approaches found considering such aspects.³ Some approaches are composed of more than one study as they were published as an evolution or

² In Sato et al. [27]'s literature review, a different and complementary classification is presented.

³ This survey covers articles published up to the year 2021.

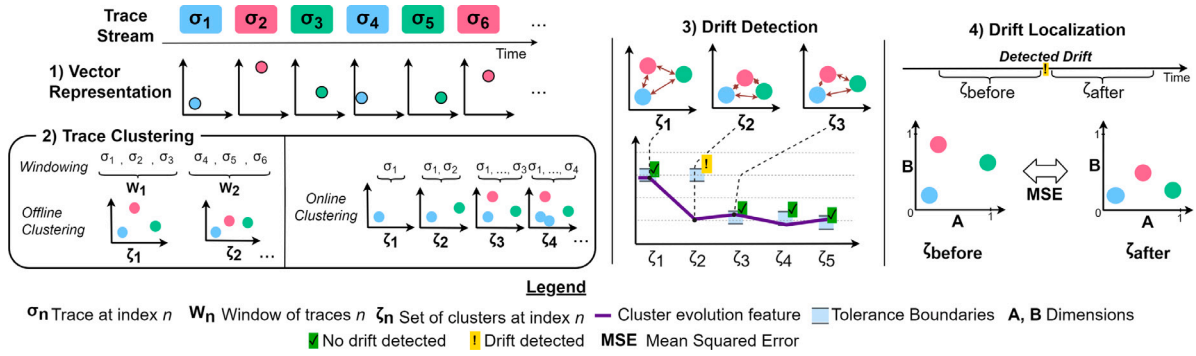


Fig. 3. Overview of the proposed approach. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.).

complement of a single approach; 40 studies were found in total. For each of the 29 approaches, we list information, techniques and strategies used to implement methods of concept drift detection, localization (Loc), characterization (Cha), and explanation (Exp). Information about the last three tasks is grouped in the *understanding methods* column. When there is more than one study published for the same approach, the *year* column refers to the first one. The *type* column shows whether the method works in an offline (batch processing) or online (data stream) scenario; for online, the * symbol means *event stream* while the lack of the * symbol means *trace stream*. The † symbol in the *category* column means the approach has been tested on public true concept drift-labeled event log. The type of concept drift detected is noted in the *detection method* column as follows: *su* for sudden, *in* for incremental, *gr* for gradual, *re* for recurrent, or *uid* for unidentifiable; according to the typology defined in the concept drift literature [5,28]. Finally, the § symbol together with the name of the detection, localization, characterization, or explanation method means that experimental results are not presented for that method.

Statistical testing-based concept drift detection methods use hypothesis testing to see whether there is a statistically significant difference in the data before and after a moment that would show a concept drift has occurred. A windowing strategy is used to extract the samples over time, before and after. Although reliable and well accepted in the literature, hypothesis testing often relies on strong assumptions about the data and hence may not account for subtle but relevant process changes. *Similarity*-based concept drift detection methods extract similarity measures from the data before and after a moment and use threshold values to see whether there is a significant difference that would show a concept drift has occurred. Comparison between before and after can be supported by a windowing or data stream strategy, or by time-related input data. Such comparisons can be supported by different strategies, including the use of trace clustering, process models, and similarity matrix. Despite their flexibility of being able to use different similarity metrics, similarity-based methods often rely on subjective threshold values defined through empirical experiments. Finally, *time series*-based concept drift detection methods extract from event logs temporal features-related measures and analyze them. Time series-based methods transform input data into a time series so that a specific time series technique is applied, such as change point detection. These methods detect significant differences over time often supported by a windowing strategy along with descriptive statistics measures (i.e., mean, standard deviation, etc.).

Like ours, 11 approaches deal with concept drift detection and localization in an integrated way. However, of those 11 approaches, only four are online. Furthermore, of those four integrated and online approaches, only one [19,29,30], which is composed of two concept drift localization methods, has been tested and evaluated through a quantitative metric on public true concept drift-labeled event logs. However, the experimental results of these two concept drift localization methods [19,30] are not explicitly listed in the respective papers, not allowing a direct comparison. Moreover, unlike our approach, Ostovar et al. [29,30] use a statistical testing-based detection and localization method, while ours uses a similarity-based detection method. The similarity-based localization method of Ostovar et al. [19] rely on a complex data structure related to tree-based process models, while ours relies on a simpler data structure related to a vector in \mathbb{R} .

3. Proposed approach

Fig. 3 depicts the approach proposed herein, with its four numbered steps. Event traces are processed together in the *trace batch* or as they arrive in the *trace stream*. In step 1, each trace is mapped into a feature vector space via a particular *trace vector representation* (using one of the trace profiles considered in this approach, as discussed in Section 3.1). In step 2, *trace clustering* is applied via one of two strategies: batch trace clustering via a windowing-based clustering method [6], or stream trace clustering by feeding each cluster directly to an online clustering method. In step 3, *concept drift detection* is applied by monitoring the variation of the clusterings over time from the extraction of features that describe the clustering at each time; a concept drift is detected when there is a significant variation between two consecutive clusterings for the measured feature. In step 4, *concept drift localization* is applied (after a concept drift is detected) by calculating the mean squared error (MSE) to analyze the clusterings before and after the detected concept drift. For both last steps, the extraction of features and the MSE are derived from clusters' structure.

Table 1
Comparison of related work — Part I.

Work	Year	Category	Type	Data representations	Detection method	Understanding methods (Localization, Characterization, and Explanation)
[2,13,31]	2011	Statistical	Offline	Relation type count, relation entropy, local window count, J-measure	Kolmogorov–Smirnov test (su/gr)	Changes for each pair of activities (Loc)
[32]	2011	Statistical	Online	Probabilistic deterministic finite automata, Alpha relations	Chi-squared, Hoeffding's inequality (su)	–
[33]	2012	Similarity	Offline	Maximal repeat (a special substring in a sequence), trace starting time	Agglomerative hierarchical clustering (gr)	–
[34]	2012	Similarity	Offline	Distance between pair of activities	Cluster cuts heuristic (su)	Activity pairs in the peaks of cluster cut graphs (Loc)
[14]	2012	Time series	Online	Trace as Parikh vectors	ADWIN on polyhedron area mass (su)	–
[35]	2015	Statistical	Offline	Event class correlation	Mann–Whitney U test, The Moses test for equal variability (su)	–
[24,36]	2015	Similarity	Offline	Stochastic similarity matrix between cases	Differences in similarity matrices, Markov clustering (su)	–
[15,37]	2015	Statistical ^b	Offline	Alpha concurrency (runs)	Chi-squared test (su/gr)	–
[19,29,30]	2016	Statistical/ Similarity ^b	Online ^a	Frequency of Alpha plus relations, Process tree model	G-test (su)	Statistic association of each Alpha plus relation with the concept drift (Loc/Cha), Edit operations in process tree model (Loc/Cha)
[11]	2017	Statistical ^b	Offline	Graph metrics from Heuristic miner	G-test (su)	Differences in the graph metrics (Loc)
[16]	2017	Similarity	Offline	Direct succession (or directly follows relation) and weak order relations	Minimum relation invariance distance, DBSCAN (su)	–
[38,39]	2018	Similarity ^b	Online ^a	Frequency of activities, Graph-distance (trace and time)	DBSCAN, DenStream (su/gr)	–
[40]	2018	Similarity	Online	Alpha relations matrix, Process model	Differences in Alpha relations matrix, metrics of process model (su/gr/in/re)	Activity pair changes (Loc) ^c /Type of concept drift based on conditions (Cha) ^c
[41]	2018	Similarity	Online ^a	Process model	Conformance checking (su/gr/in/re)	Type of concept drift based on conditions (Cha)
[42]	2019	Time series	Online ^a	Time interval between activities	Exponentially moving z-score (su/in/re)	Graphs for each pair of activities (Loc)

^a Means *event stream* (the lack of ^a means *trace stream*).

^b Means the approach has been tested on public true concept drift-labeled event log.

^c Means that experimental results are not presented for that method.

3.1. Trace profile

The nature of the data provided by the event logs allows the process to be analyzed from different perspectives, such as control-flow (related to the order of activities), organizational (related to resource organization), case (related to trace characterization), and performance (related to time performances) [7]. These perspectives can be used to generate different trace vector representations, which are known as *trace profiles* [17,21].

Each trace profile is based on information associated with or derived from different event log attributes, such as activity, timestamp, resource, and other domain-specific data [21]. In our approach, the trace profiles are based on: *activity* (directly associated with the activity attribute) and activity *transition* (derived from the activity attribute to identify pairs of activities where one is directly followed by the other in an event trace⁴). Each unique value associated with or derived from the chosen attributes becomes a dimension in the vector. For example, one dimension is created for each activity name in the event log, for the activity profile, or for each transition between activities in the event log, for the transition profile.

The value assigned to each dimension can result from different strategies, three of which are applied in our approach: *occurrence*-based strategy, a binary count-based strategy, in which only the presence or absence of the value – associated with the vector dimension – in the trace is considered, i.e., *1* whether it occurs or *0* otherwise [17,21]; *frequency*-based strategy, a non-binary

⁴ In fact, this profile would more correctly be called a *directly-follows relation* profile rather than a *transition* profile, as one activity directly followed by another in an event trace does not necessarily represent a transition in a process [7]. However, we kept the nomenclature used by previous authors [17,21].

Table 2
Comparison of related work — Part II.

Work	Year	Category	Type	Data representations	Detection method	Understanding methods (Localization, Characterization, and Explanation)
[43]	2019	Statistical	Offline	Extended dynamic Bayesian network	Kolmogorov–Smirnov test (uid)	–
[44]	2019	Time series	Online ^a	Directly follows relations	ADWIN technique (su)	–
[45,46]	2019	Similarity	Offline	Metrics of process model, trace and activity levels	Process model comparison (uid)	–
[47–50]	2019	Time series ^b	Offline	DECLARE constraints confidence	Pruned exact linear time method (su/gr/in/re)	Concept drift chart and extended directly-follows graph (Loc)/Type of concept drift using concept drift chart and concept drift measures (Cha)
[51]	2020	Similarity	Offline	Trace and time as stochastic languages	Earth mover’s distances (su)	–
[52]	2020	Similarity ^b	Offline	Direct succession (or directly follows relation) relations	Local completeness (su)	–
[53]	2020	Similarity	Offline	Dynamic networks	Pattern-based change detection (su)	Maximal emerging subtrees (Loc)
[54]	2021	Time series	Offline	Several measures extracted over time	Pruned exact linear time method (su)	Cause-effect analysis with Granger-causality (Exp)
[55]	2021	Statistical	Offline	Directly-follows relations	G-test, adjusted standardized residual (su)	Directly-follows relations changes (Loc) ^c
[56]	2021	Time series	Offline	Exclusive choice splits from process model	Pruned exact linear time method (su)	Concept drift for each exclusive choice split (Loc)
[57]	2021	Similarity	Online ^a	Z-scoring of temporal deviation signature	Visual monitoring based on OPTICS clustering (uid)	–
[58]	2021	Similarity	Online	Geodesic distance between non-conforming traces	Clustering based on Local Outlier Factor (re)	–
[59]	2021	Similarity ^b	Online	DFG with frequencies of activities and paths	Node and edge similarity metrics from the DFG (su)	Visually from DFG (Loc)
[60]	2021	Statistical	Offline	Gamma mixture models (GMMs) of process duration	Multinomial, log-likelihood ratio and Kolmogorov–Smirnov tests (su)	–
Our approach	2022	Similarity ^b	Online	Activity and transitions trace vector profile	Variation from clustering evolution (su)	Rank and intensity of the entities involved in the concept drift (Loc)

^a Means *event stream* (the lack of ^a means *trace stream*).

^b Means the approach has been tested on public true concept drift-labeled event log.

^c Means that experimental results are not presented for that method.

count-based strategy, in which the number of times the value – associated with the vector dimension – appears in the trace is counted [17,21]; and *order*-based strategy, in which the position at which the value – associated with the vector dimension – first appears in the trace is divided by the length of the trace to generate a normalized ascending order, and 0 is assigned to vector dimensions whose associated values are not present in the trace.

Considering the combination of both the type of information and the way of assigning values to vector dimensions, four trace profiles are applied in this approach: *activity-occurrence*, *activity-frequency*, *activity-order*, and *transition-occurrence*. Each trace profile is conceptually based on a specific view of the process underlying the event log and is, therefore, inherently limited to that view. Consequently, each trace profile can only represent specific types of process behaviors. For example, the *activity-occurrence* trace profile allows to only identify whether or not the activity is present in the trace without, therefore, containing information about loops and repeated occurrence of activities. As a result, the *activity-occurrence* trace profile is not suitable for detecting concept drifts when the change in process behavior involves loops. On the other hand, concept drifts involving loops would be detectable through the *activity-frequency* trace profile by counting the number of occurrences of activity in the trace. The activity-based trace profiles was chosen because of its simplicity (it generates a low-dimensional feature vector space), while the transition-based trace profile was chosen to allow the representation of a greater number of process behavior change patterns. The *activity-order* profile can represent all process behavior change patterns, but from a different perspective. In this profile, each activity is numbered considering the sequence in which they occur, so that the set of numbers is normalized according to the trace length. For activities missing from the trace, the value 0 is set on the vector. For example, given the set of activities A, B, C, and D; a trace <A, C, B> would result in the <0.33, 1, 0.67, 0> vector.

Prior knowledge of all distinct values of the chosen attributes is proposed to define the maximum vector dimension needed to build a standard feature vector space. This standard vector space is required to allow comparison of results for different parts of the event log comprising distinct traces. Otherwise, a different vector space might be built for each trace in the event log.

3.2. Trace clustering

Once the traces are mapped into the feature vector space, similar traces are grouped into clusters that can be used to represent a process behavior model. The key challenge in this step is dealing with traces arriving in an online data stream. For this, two strategies are proposed in the following subsections: carrying out a batch trace clustering using a fixed data windowing strategy and a traditional clustering method (Section 3.2.1), and directly carrying out a stream trace clustering through an inherently online method (Section 3.2.2).

3.2.1. Batch trace clustering

One of the simplest ways to handle an online data stream in general data mining is through windowing over batch data. By applying windowing to process batch data, only data within the window is considered, while data outside the window (or older data) are not considered during analysis [61]. As a result, traditional clustering methods can be applied on each window to independently cluster the most recent data. We employ the k-means++ clustering method [62], using a batch (windowed) strategy, in the approach proposed herein for two main reasons: its simplicity and wide application to cluster traces in other process mining tasks besides concept drifts; and its ability to provide centroid-based cluster models, which are essential for carrying out the concept drift detecting and localization steps in our approach. We refer to this strategy here as window-based k-means++.

The trace stream is processed using the landmark window model, which comprises processing windows of a fixed number of traces. Windows are labeled W_i , where i is the position of the most recent trace in the batch. All traces in the windows are individually mapped into the same feature vector space to allow their comparisons. We assume each window contains a sample of traces representing the process behavior at its corresponding time. Thus, clusters are expected to be stable along the trace stream in window sequences without concept drifts, as clusters abstract the process behavior. On the other hand, clusters are expected to show variations between different windows in response to concept drifts that change the process behavior.

3.2.2. Stream trace clustering

Stream clustering algorithms are adaptations of traditional clustering methods to work directly with data stream naturally, without requiring the use of data windows, like the one presented in Section 3.2.1. Accordingly, stream clustering algorithms update existing clusters by accounting for new data points in the existing cluster model, identifying changes in clustering structure, such as cluster merge, split, emergence, disappearance, expansion, and shrinkage [63]. In this study, we investigated the effectiveness of this type of method in detecting concept drifts and compared it to batch trace clustering.

In the approach proposed herein, we employ the CluStream stream clustering method [64]. CluStream was chosen because it is one of the reference algorithms for directly dealing with an online data stream due to its ability to consider different time horizons [65]. CluStream creates and maintains multiple microclusters by allocating data points to them or creating new ones, and deleting or merging others. Microclusters do not store data points, but summary statistics of the data points (i.e., the sum of the data values and the sum of the squares of the data values allocated to a microcluster, the sum of the arrival timestamps, and the sum of the squares of the arrival timestamps of each stream data point), arranged in cluster feature vectors. As these features have additive properties, they can be efficiently updated for each stream data point, i.e., online. In addition to microclusters, CluStream creates macroclusters through a traditional clustering method that processes cluster feature vectors as pseudo-data points, considering the number of output clusters k and a history of length h (with h being the time horizon), both specified by the user. At each execution of the traditional clustering method, the initialization seed is replaced by the centroids of the clusters obtained in the last iteration. Furthermore, to consider the user-defined time horizon (h), the method deals with microclusters created in two moments: the current one and the one corresponding to the current one minus the length of h . Such temporal processing is possible due to the pyramidal time frame concept, which organizes microclusters temporally, allowing algorithms to retrieve information from different time horizon perspectives. In this scenario, each trace in the stream is processed once by the clustering method.

3.3. Concept drift detection

Trace clustering provides an abstraction of clustered data representing behavior profiles of the process underlying the event log. As concept drift is a problem related to concept change and evolution, one or more behavior profiles change and the corresponding clusters need to be tracked for variations that show concept drifts occurring. To make this tracking possible, Section 3.3.1 presents a set of features to be extracted that can describe the clusterings. Then, Section 3.3.2 describes the proposed method to detect concept drifts based on the extracted features.

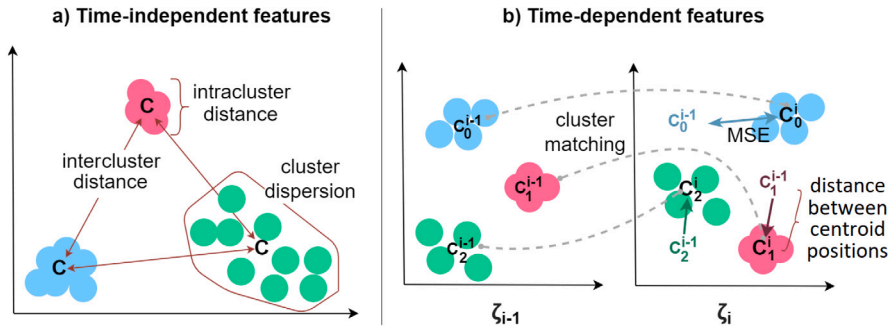


Fig. 4. Examples of time-independent and time-dependent features extracted. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.).

Table 3
Clustering variation features.

Feature	Aggregations	Type
Intracluster distance	avg, std	Time-independent
Intercluster distance	avg, min, max, sum	Time-independent
Squared sum of centroids	–	Time-independent
Silhouette coefficient [68]	–	Time-independent
Davies–Bouldin index (DBI) [69]	–	Time-independent
MSE	avg, sum, count_non_zero	Time-dependent
Distance between centroid positions	avg	Time-dependent

3.3.1. Tracking clustering variations

Different types of concept drifts can affect trace clustering in several ways, such as changing centroid positions or merging clusters. Frameworks capable of identifying changes in clustering over time were introduced by Spiliopoulou et al. [66] and Oliveira and Gama [67]. These frameworks can identify external transitions, such as cluster merge, split, emergence, and disappearance, as well as internal transitions, such as cluster expansion, shrinkage, and density variation.

Formally, a clustering $\zeta_i = \{C_1^i, C_2^i, \dots, C_j^i, \dots, C_k^i\}$ is a set of k clusters C obtained at the end of each run i of the clustering method. Based on Spiliopoulou et al. [66] and Oliveira and Gama [67], two types of clustering variation features are extracted (see Fig. 4 for illustrative examples): (a) time-independent, which consider only the properties of the current clustering; and (b) time-dependent, related to the comparison of two consecutive clusterings (ζ_{i-1} is the previous clustering; ζ_i is the current clustering).

Our approach relies on centroid-based clustering algorithms (as k-means++ and CluStream), which represent a cluster C_j by a centroid. A centroid is the central point of a cluster and is defined here as the average of all data points that compose the cluster. Individual features are calculated for each cluster, based on its centroid, to extract time-independent features, such as intracluster and intercluster distances. Table 3 lists the features and aggregations used in our approach.

For the intracluster distance, we consider the average pairwise Euclidean distance for all data points in the cluster and for the intercluster distance, we apply the Euclidean distance between all k centroids in a clustering ζ_i . As some features are related to each cluster individually, we aggregate them to get a single value per clustering ζ_i .

Time-dependent features are based on comparing a cluster C_j^{i-1} with its matching cluster C_j^i in the following clustering. The choice of the matching pairs of clusters is treated as an assignment problem and solved using a heuristic based on the Hungarian method [70]. As our approach always runs clustering with a fixed k , a 1:1 matching between clusters is always obtained and hence no cluster merging or splitting occurs. The MSE measures the variation of the matching centroids. For the distance between two consecutive matching clusters, the Euclidean distance measures the variation occurred in the spatial position between C_j^{i-1} and C_j^i .

In addition, these features were adapted to allow their extraction for a batch non-clustering strategy⁵ (which was also applied, cf. the experiments presented in Section 4), in order to enable the comparison of its results with the other two clustering-based strategies. In the batch non-clustering strategy, we consider a fixed data windowing strategy and all data points (traces) in a window to extract the time-dependent features and the time-independent features. For features with cluster-internal measures, such as *intracluster distance*,⁶ all data points (traces) were assumed to belong to the same cluster.

3.3.2. Concept drift detection method

Our concept drift detection method consists of verifying whether the current value of a given feature has undergone a significant variation. As each feature value represents the clustering of a significant number of traces, we assume it should remain stable

⁵ Although this is not a clustering-based strategy, we also refer to it as batch processing due to the use of trace windows.

⁶ In this case, we did not use features based on intercluster distance.

according to the process behavior underlying the traces when there is no concept drift. The proposed method iterates over any of the clustering variation features over time and, in each iteration, the feature value is compared to a dynamically estimated tolerance boundary. A feature value outside the tolerance boundary represents a significant behavior variation from what was previously measured. To define what is a significant variation, we estimate upper and lower limits for a tolerance boundary based on the average (i.e., a rolling average) and standard deviation of the feature previous values.

A detailed description of the concept drift method is shown in Algorithm 1. The algorithm has four input parameters: v series of values of the chosen feature obtained either with clustering over trace windows for batch trace clustering or after each new trace has been processed for stream trace clustering; rw size of the rolling average window used to analyze the behavior contained in v , i.e., how many values of the analyzed feature should be considered; θ threshold on how many standard deviations to tolerate to consider normal behavior in v , adapting the size of the tolerance boundary considering the rolling standard deviation, to avoid the detection of false positives concept drifts in more unstable segments of the event log; and mb minimum tolerated variation, based on a percentage of the rolling average, for a concept drift to be considered in v , especially useful to deal with noise and hence avoid false positives in a scenario with low standard deviation. The rolling average and the standard deviation represents the average and the amount of variation in process behavior over consecutive trace windows. Parameter rw affects the smoothness in the feature values (v), making the detection method more sensitive to anomalies and minor changes if set to lower values. Higher values of θ lead to wider limits of the tolerance boundary making concept drift detection less frequent, while lower values of θ lead to narrow limits of the tolerance boundary leading to more false positives. The concept drift localization method is called after each concept drift detection. As an output, the detection method provides a list of the clustering indexes where concept drifts were detected.

Algorithm 1 Concept drift detection algorithm

```

1: Input
2:    $v$    Vector of feature values
3:    $rw$   Rolling average window size
4:    $\theta$    Standard deviation tolerance
5:    $mb$   Minimum percentage for lower and upper limits of the tolerance boundary
6: Output
7:    $d$    List of clustering indexes of detected concept drifts
8: procedure CONCEPT_DRIFT_DETECTION( $v, rw, \theta, mb$ )
9:    $d = \emptyset$ 
10:   $buffer = \emptyset$ 
11:   $rolling\_avg = none$ 
12:   $rolling\_std = none$ 
13:   $clustering\_index = 0$ 
14:  for each  $feature\_value \in v$  do
15:    if  $size(buffer) = rw$  then
16:       $lower\_toler = \min(rolling\_avg - (rolling\_std * \theta), (1 - mb) * rolling\_avg)$ 
17:       $upper\_toler = \max(rolling\_avg + (rolling\_std * \theta), (1 + mb) * rolling\_avg)$ 
18:      if  $feature\_value < lower\_toler$  or  $feature\_value > upper\_toler$  then
19:         $d.append(clustering\_index)$ 
20:         $buffer = \emptyset$ 
21:         $concept\_drift\_localization(C[clustering\_index - 1], C[clustering\_index])$ 
22:        //Where C is a function that returns the centroids of clusters
23:        //A concept drift will only be detected and located when clustering_index is > 1.
24:      end if
25:       $buffer.pop(0)$ 
26:    end if
27:     $buffer.append(feature\_value)$ 
28:     $rolling\_avg = average(buffer)$ 
29:     $rolling\_std = std(buffer)$ 
30:     $clustering\_index += 1$ 
31:  end for
32: end procedure

```

Algorithm 1 iterates over each value of the chosen feature in v . For each iteration, the feature value is appended to $buffer$ [ln 27]. Current $rolling_avg$ and $rolling_std$ are assigned in each iteration based on the updated $buffer$ [lns 28-29] to be used if a concept drift test needs to be carried out in the next iteration [lns 15-26]. When $buffer$ is full, i.e., it reached the window size rw [ln 15], a concept drift test is carried out. First, the tolerance boundary (lower and upper) are calculated [lns 16-17]. Then, this tolerance boundary is used to test whether the current feature value should be considered a concept drift [ln 18]. A concept drift is detected if the current feature value is outside the upper and lower limits of the tolerance boundary calculated based on $rolling_avg$ and

rolling_std from the previous iteration. If a concept drift is detected, d receives the clustering index of the previous feature [Ln 19], *buffer* is emptied to discard old behavior [Ln 20], and the localization method is triggered [Ln 21] (as shown in Section 3.4). When *buffer* is full, even with no concept drift detected, the earliest value is removed [Ln 25] for appending a new feature value [Ln 27].

Algorithm 1 is subject to an inherent delay in relation to the clustering trace window index or to the time horizon to which the concept drift refers, as it demands that a certain number of traces after the concept drift has been processed to be able to detect the concept drift, otherwise it could be the case of an anomaly or outlier. Trace window size, time horizon, and rolling average window size influence the length of the deviation detection delay.

3.4. Concept drift localization

Our concept drift localization method works in the sequence of the concept drift detection method, in an integrated approach, so as to leverage the clusters under analysis at the moment to solve both tasks at once, i.e., detect and localize concept drifts. When a concept drift is detected, the clustering information from the previous iteration is referenced so that the localization of the given drift is executed at the current interaction of the detection method (cf. [Ln 21] in Algorithm 1). Thus, the concept drift localization is able to consider clustering situations before and after the detected concept drift.

The concept drift localization method compares the clustering situations before and after the detected concept drift (i.e., ζ_{before} and ζ_{after}) to identify the process elements involved in the concept drift. The localization method takes ζ_{before} and ζ_{after} as input and compares the variation of the centroid positions for each pair of matching clusters from ζ_{before} and ζ_{after} . As a concept drift can affect only a specific part of the process, then the comparison of ζ_{before} and ζ_{after} considers the possible variation in the positioning of the individual centroids of all the clusters of each of ζ_{before} and ζ_{after} . A centroid matching between ζ_{before} and ζ_{after} is carried out identically to what is done for time-dependent features (cf. Section 3.3.1).

To compare the positioning variation between ζ_{before} and ζ_{after} , the mean squared error (MSE) is calculated for each dimension of the cluster centroids before and after the concept drift. Eq. (1) shows the calculation of the MSE for each dimension d , where ζ_{before} is the clustering situation before the detected concept drift, ζ_{after} is the clustering situation after the detected concept drift, k is the number of clusters, c is the centroid of a cluster, and $c_{j,d}$ is the centroid value of cluster j in dimension d .

$$MSE(d) = \frac{1}{k} \sum_{j=1}^k \left(c_{j,d}^{\zeta_{before}} - c_{j,d}^{\zeta_{after}} \right)^2 \quad (1)$$

As an output, the concept drift localization method provides an ordered list of process elements – represented by the corresponding feature vector dimensions – accompanied by their respective MSEs representing their chances of being involved in the concept drift. In our approach, each dimension of the feature vector space represents either an activity or a transition between activities (depending on the trace profile used). With this information, one can identify which dimensions are related to the concept drift and to what degree. Thus, comparison by each dimension means evaluating the difference in process behavior (before and after) with respect specifically to each of the activities (or to each of the transitions between activities) of the process model. Obtaining high MSE values in a given dimension of the centroid shows the corresponding activity or transition behaves differently before and after the concept drift. The different MSE values calculated for each dimension represent the degree of change in behavior of each corresponding activity or transition. Such information can be used by domain experts to obtain descriptive knowledge about the concept drifts found.

This proposed concept drift localization method relies only on the centroids resulting from the clusterings and thus is independent of the clustering strategy applied for trace clustering (either batch or stream) to support concept drift detection. Despite this, there is a difference in how to identify ζ_{before} and ζ_{after} to apply in Eq. (1) for each trace clustering strategy. In both cases, ζ_{after} is always the clustering situation immediately after the moment when the concept drift was detected, which corresponds to the iteration of the detection method in which the concept drift was detected. However, ζ_{before} is different in each case. For batch trace clustering, ζ_{before} is the clustering situation immediately before the moment when the concept drift was detected. As for stream trace clustering, the situation immediately before clustering does not fully capture the process behavior before the drift as the difference is only one trace. Thus, it required us to adapt the reference comparison window according to the CluStream time horizon parameter h , in which we consider the clustering situation h iterations before the drift as ζ_{before} to better represent the process behavior before the drift.

4. Experiments with synthetic event logs

In this section, we present the results of applying our approach to synthetic event logs. Distinct change patterns representing concept drifts were artificially inserted during the construction of these event logs so that they represent a ground truth for testing solutions. Our approach was tested considering different settings for the parameter values of the synthetic event logs used. Two sets of synthetic event logs were used — one (named here as *loan application process*) based on an elementary process model with various change patterns, presented by Maaradji et al. [71], and another (named here as process with *high trace variability and noise*) based on a more complex process model that simulates a high trace variability and noisy scenario, presented by Ostovar et al. [19]. First, for the loan application process, we present additional discussions to illustrate how our batch and stream trace clustering strategies work. We compared the effectiveness of the batch trace clustering strategy (using window-based k-means++) against the stream trace clustering strategy (using CluStream). Furthermore, we used a batch non-clustering strategy, in which no cluster is applied within a trace window, to investigate the actual usefulness of trace clustering in concept drift detection and localization.

4.1. Experiments' setup

For the batch trace clustering strategy, the window-based k-means++ runs for 10 iterations with different initial centroids for each trace window, and the best result in terms of quantization error⁷ is selected as the result of the clustering step. As for the stream trace clustering strategy, CluStream is fed with each trace individually (i.e., in a stream manner), and features are extracted from the microclusters and macroclusters after processing each trace. The batch non-clustering strategy calculates the time-dependent and time-independent variation features using the landmark window model, performing non-clustering, considering only the traces in the window. For example, when calculating *avg_dist_between_centroids*, the average distance between all traces in a window is calculated using the centroids of each cluster instead of using the traces themselves; or else when calculating *avg_diff_centroids*, the average of all traces is calculated as a single centroid in each trace window, and then the distance is calculated between two centroids, one in each of the subsequent trace windows.

The experiments explored the impact on results of different values for specific parameters and the search for the best combination of parameter values. Two minimum sets of parameters were considered: one related to the trace clustering step, which is the basis for both the concept drift detection and localization methods; and another related exclusively to the concept drift detection method. For the first set, the trace clustering parameters rely on the specific strategy applied, as follows:

- Batch trace clustering strategy: *ws* sets the fixed size for the landmark window model, i.e., sets the number of traces to be considered in each trace window; and *k* sets the number of clusters to be obtained by window-based k-means++.
- Stream trace clustering strategy: *m* sets the number of microclusters to be maintained; *h* sets the length of the time horizon that controls the expiration of microclusters; *t* sets the size of a microcluster's radius so that new data points can be appended to it; and *km* sets the number of macroclusters to be obtained through k-means fed with the microclusters.
- Batch non-clustering strategy: *ws* sets the fixed size for the landmark window model, i.e., sets the number of traces to be considered in each trace window.

As for the second set of parameters, the concept drift detection parameters are composed of *rw*, θ , and *mb*, as described in Section 3.3.2.

Both sets of synthetic event logs are accompanied by ground truth information for the concept drift detection task, which shows the specific times when concept drifts occur. Such ground truth information makes it possible to evaluate the effectiveness of the concept drift detection method of our approach through the F-score measure by accounting for the balance among true positives, false positives, and false negatives obtained in the attempts to detect concept drifts. To classify a drift detection as correct (i.e., a true positive) in relation to ground truth information, we employed a maximum lag period of two trace windows for both batch trace clustering and batch non-clustering strategies, and two time horizons (for the stream trace clustering strategy).

The first set of synthetic event log is accompanied by ground truth information for the concept drift localization task in the form of process models used to derive event log snippets referring to the control-flow of the base process and the control-flow of the process altered according to a change patterns. Thus, we developed a heuristic to evaluate our localization method's effectiveness. As for the second set of event logs, there is no such a ground truth information, and so the effectiveness of the concept drift localization method of our approach is evaluated based on an analysis of process models discovered before and after the occurrence of the detected concept drifts.

To evaluate the effectiveness of the MSE-based concept drift localization method, given the presence of ground truth information, we propose a *ad-hoc* approach inspired by ROC curve analysis. Our localization method relies on MSE to calculate the distance between centroids (cf. Fig. 3(4)), capturing the extent of centroid displacement across each dimension representing an activity or transition. According to this localization method, the calculated MSEs serve as an indicative measure of the dimensions implicated in the concept drift: the dimensions with the greatest displacements (i.e., higher MSEs) are the most likely to be involved in the concept drift. Given a vector of calculated MSEs \bar{c} , our *ad-hoc* approach first systematically assesses a set of thresholds, varying in the range bounded by the minimum and maximum values in \bar{c} : values in \bar{c} greater than or equal to a given threshold suggest that the respective dimension (activity or transition) is involved in the concept drift. Subsequently, for each threshold, the rate of correctly indicated dimensions involved in the concept drift (true positive rate — TPR) and the rate of incorrectly indicated dimensions involved in the concept drift (false positive rate — FPR) are calculated. Based on the TPR and FPR sets calculated for all thresholds, a curve on the FPR \times TPR graph is constructed, and the area under this curve is calculated. The value of this area serve as a heuristic to evaluate the effectiveness of the localization method. The heuristic posits that a sufficiently high area value indicates the existence of at least one threshold capable of correctly indicating a reasonable number of dimensions involved in the concept drift without incorrectly indicating an unreasonable number of dimensions involved in the concept drift. For a conservative analysis, we consider a minimum area under the curve of 0.875. Thus, areas under the curve in the range [0.875, 1.0) strong suggest the presence of at least one threshold with a positive trade-off between correct and incorrect indications of dimensions involved in the concept drift. On the other hand, an area under the curve equal to 1 signifies the presence of at least one threshold indicating the entirely correct set of dimensions involved in the concept drift (i.e., 100% TPR and 0% FPR).⁸

⁷ Quantization error: the minimum sum of squared distances from each data point to its nearest centroid.

⁸ For instance, consider four dimensions representing four activities (*A, B, C, D*), with two activities involved in the concept drift (*A, D*). Consider also: the calculated MSE values $\bar{c} = (1.25, 0.75, 0.01, 0.69)$; values greater than or equal to the threshold indicate "involvement in the concept drift" (positive — *P*); values less than the threshold indicate "non-involvement in the drift" (negative — *N*). Assuming four (sufficient) cutoff thresholds: (1.25, 0.7, 0.5, 0.01), the following

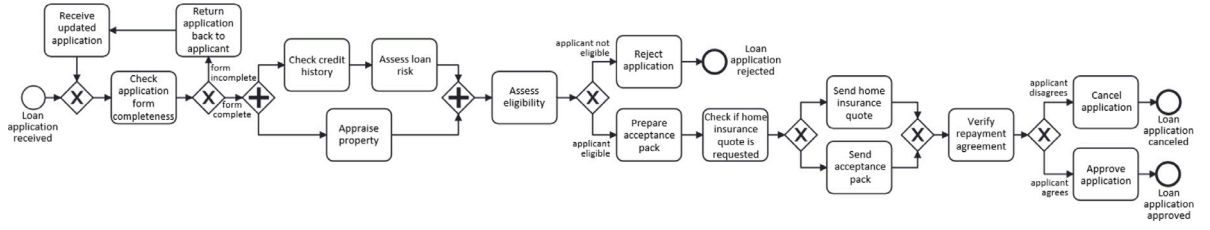


Fig. 5. Loan application base process model.
Source: Adapted from Maaradji et al. [71],
Dumas et al. [72].

Table 4

Loan application — change patterns present in event log and support by trace profiles of our approach.

ID	Description	Category	Trace profile			
			Activity			Trans
			Occur	Freq	Order	Occur
cm	Move fragment into/out of conditional branch	I	✓	✓	✓	✓
pm	Move fragment into/out of parallel branch	I			✓	✓
re	Add/remove fragment	I	✓	✓	✓	✓
rp	Substitute fragment	I	✓	✓	✓	✓
cp	Duplicate fragment	I		✓	✓	✓
sw	Swap two fragments	I			✓	✓
cd	Synchronize two fragments	R			✓	✓
pl	Make two fragments parallel/sequential	R			✓	✓
cf	Make two fragments conditional/sequential	R	✓	✓	✓	✓
cb	Make fragment skippable/non-skippable	O	✓	✓	✓	✓
lp	Make fragment loopable/non-loopable	O		✓	✓	✓
fr	Change branching frequency	O	✓	✓	✓	✓

4.2. Loan application process

This first set of 54 synthetic event logs was created from the *loan application* base process model [15,71], which has (cf. Fig. 5) 15 activities, one start event, three end events, one loop structure, as well as some parallel and alternative branches.

The event logs were created to simulate different concept drifts resulting from different types of process control-flow changes. The base process model was altered according to change patterns commonly identified in process models, such as adding, removing, or swapping fragments. Table 4 lists the control-flow changes entered in the event logs with their corresponding categories. The changes are grouped into three categories: insertion (I), resequentialization (R), and optionalization (O). To simulate more intricate change patterns, there are also composite event logs generated by applying one change for each category in a nested manner (IOR, IRO, OIR, ORI, RIO, and ROI). For example, the IRO change pattern was obtained by first adding a new activity (category I), then making that activity parallel (R), and finally putting it in a loop structure (O).

Five event logs were initially generated from the base process model, and an additional five event logs were generated from each of the 18 altered process models. These event logs were created using BIMP,⁹ a business process simulation tool employing a *play-out* style strategy [7]. Subsequently, 18 event logs were created by interleaving the five event logs generated from the unaltered base process model with five event logs generated from process models altered to incorporate specific change patterns, ensuring that each event log was used only once. Each resulting event log comprises nine concept drifts. This entire procedure was carried out by considering base event logs and altered event logs with 500, 750, and 1000 cases, resulting in 18 event logs with 5000 cases, 18 event logs with 7500 cases, and 18 event logs with 10,000 cases [15]. This type of procedure generates event logs with sudden-type drifts.

responses for the involvement of dimensions in the concept drift are obtained for each threshold, respectively: $\langle P, N, N, N \rangle$, $\langle P, P, N, N \rangle$, $\langle P, P, N, P \rangle$ and $\langle P, P, P, P \rangle$. When comparing these responses with the ground truth $\langle P, N, N, P \rangle$, the corresponding points on a graph (rate of incorrect indications \times rate of correct indications) are, respectively: (0.0, 0.5), (0.5, 0.5), (0.5, 1.0), and (1.0, 1.0). In this case, the area under the curve is 0.75, indicating that the trade-off between correct and incorrect indications is not acceptable as there is no threshold capable of correctly indicating a reasonable number of dimensions involved in the concept drift without incorrectly indicating an unreasonable number. If we considered the calculated MSE values $\bar{e} = (1.25, 0.55, 0.01, 0.80)$, our *ad-hoc* approach would encounter the following points to the same thresholds: (0.0, 0.5), (0.0, 1.0), (0.5, 1.0), and (1.0, 1.0). In this case, the area under the curve is 1, indicating that for one of these four thresholds it was verified that the method was able to correctly locate the dimensions involved in the concept drift.

⁹ <https://bimp.cs.ut.ee>

Table 5
Loan application — parameters used in the experiments.

Batch trace clustering		Stream trace clustering		Batch non-clustering	
Par	Values	Par	Values	Par	Values
<i>(a) trace clustering parameters</i>					
ws	{50, 75, 100, 125, 150, 175, 200, 250}	m	{15, 30, 50}	ws	{100, 125, 150, 200}
k	{2, 3, 6}	h	{50, 150, 200}		
		t	{2, 25, 50}		
		km	{2, 3, 6}		
<i>(b) concept drift detection parameters</i>					
rw	{3, 4}	rw	{150, 200}	rw	{3, 4}
θ	{1.0, 1.5, 2.0}	θ	{0.1, 0.5, 1.0, 1.5, 2.5, 3.5}	θ	{1.25, 1.5, 1.75, 2.0}
mb	{0.3%, 0.5%, 0.7%, 1.0%}	mb	{0.01%, 0.1%, 0.3%, 1.0%, 3.0%}	mb	{0.3%, 0.5%, 0.7%, 1.0%}

Table 4 also shows which change patterns can be represented by each of the four trace profiles used in our approach (marked with a ✓ sign). Both the *activity-order* and *transition-occurrence* trace profiles can represent all simple and composite change patterns, which cannot be done by *activity-occurrence* and *activity-frequency*. For example, as *activity-occurrence* can identify only whether or not an activity is present in the trace, then it cannot represent the existence or non-existence of loops and hence not concept drift change patterns involving loops either (see change pattern *lp*).

Table 5 lists the parameters used in the experiments with this set of event logs, arranged by each trace clustering strategy. Both types of parameters are listed, i.e., parameters used by the trace clustering method and parameters used by the concept drift detection method. The values were chosen empirically through exploratory pre-experiments. For ws , we chose values that, while not very large, could contain a significant number of traces. For k , we applied the elbow-knee method to select appropriate values. The values for the other parameters were chosen by analyzing the results obtained for this set of event logs in the exploratory pre-experiments, and interpreting the time-dependent and time-independent features behavior over time and their relationship with the known concept drifts in the event logs.

4.2.1. Effect of trace clustering on the concept drift analysis

We apply trace clustering to analyze concept drift based on the hypothesis that isolating distinct process behaviors into groups may facilitate the perception of behavior changes. One can explore the effect of applying trace clustering on the concept drift detection task by studying which process behavior profiles are clustered, as well as the results from extracting the time-dependent or time-independent features using information about these profiles.

To illustrate how the trace clustering method clusters different parts of the process behavior and its potential to influence the concept drift analysis, we take a magnifying glass and investigate the specific traces placed in each cluster. This analysis is carried out based on the process models discovered¹⁰ for each cluster¹¹ separately. Fig. 6 shows the results of a visual inspection regarding the differences before and after a known occurrence of a concept drift based on change pattern *cb* in which some activities are made skippable. In this figure, white backgrounds refer to process instances executed before the concept drift occurrence, while green and yellow backgrounds refer to process instances executed after the concept drift occurrence. The green background shows no changes were found in the process, while the yellow background shows changes were found in the process and the red circle highlights such changes. For this case where $k = 2$, the first cluster (Cluster 0) includes only traces that end with *loan application rejected*, while the second one (Cluster 1) includes the rest of traces that can finish with either *loan application canceled* or *loan application approved*. There is no difference between the Cluster 0's process models. As for Cluster 1, there are differences between the process models. The concept drift occurrence is characterized by the activities *Prepare Acceptance Pack* and *Check if home insurance quote is requested*, which become optional since a XOR gateway before each of them is included in the process model.

In this example, trace clustering isolated all process instances involved in the concept drift in Cluster 1 and, in effect, allowed feature extraction to better absorb the change information. Fig. 7 shows a comparison among the feature series using two clustering strategies and without using clustering. The concept drifts can be more easily perceived as a recurring pattern when using the clustering strategies (green and yellow lines) as opposed to the batch non-clustering strategy (red line). In the latter, the feature series is not as stable and the recurring pattern is not so well-defined. The nine pairs of log fragments before and after concept drift are likely to differ from each other due to the procedure followed to generate the corresponding event log (cf. Section 4.2), introducing variability in scenarios for concept drift detection. These potential variabilities arise from factors such as the presence of different variants, varying frequencies of the same variant, and diverse orders of occurrence among different variants, which may represent additional concept drifts (i.e., those not created by the aforementioned generation procedure). These scenario variabilities explain why concept drift is detected at certain points but not at others. In particular, the batch trace clustering strategy exhibits greater robustness in handling these variabilities compared to the stream trace clustering and non-clustering strategies.

¹⁰ The Inductive Miner method proposed by Leemans et al. [73] was used.

¹¹ The clustering algorithms were carried out aiming to find two clusters.

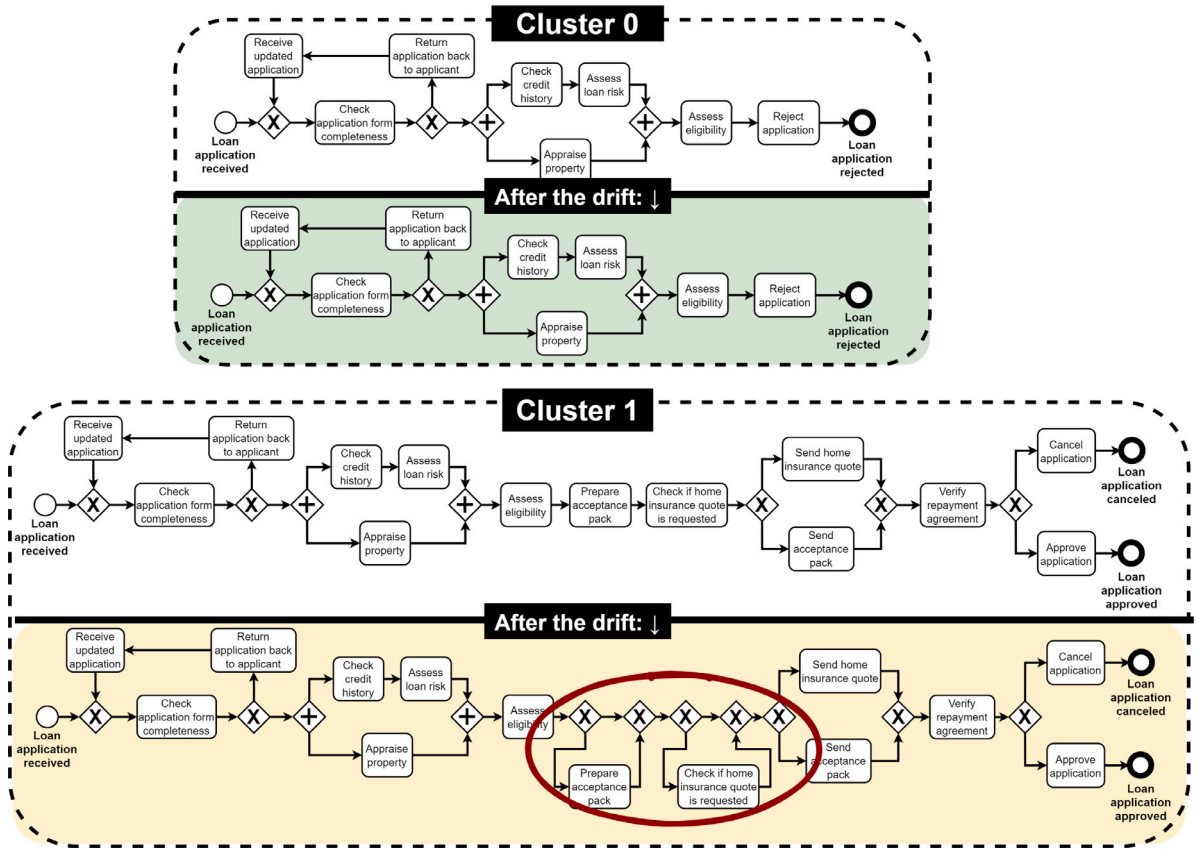


Fig. 6. Loan application — process model discovery applied to each cluster before and after a known occurrence of a concept drift (for change pattern *cb*). Batch trace clustering strategy with $k = 2$ and $ws = 100$. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.).

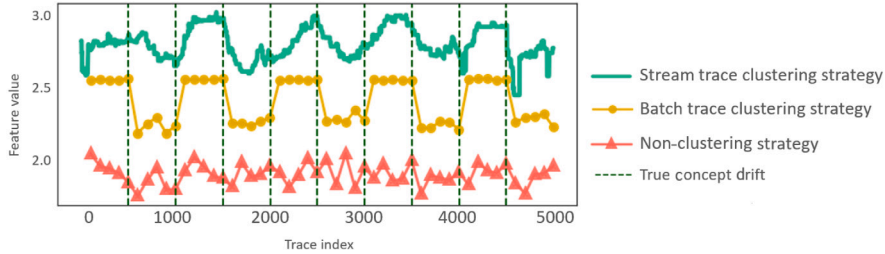


Fig. 7. Loan application with change pattern *cb* — variation of the average distance between centroids feature, extracted by applying two trace clustering strategies and a batch non-clustering strategy. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.).

4.2.2. Illustration of the concept drift detection method

Although each trace clustering strategy (i.e., batch and stream) relies on different execution dynamics, they both capture the same trend for the overall concept drift behavior. Concept drift detection results can be seen in Fig. 8(a) for the batch trace clustering strategy and in Fig. 8(b) for the stream trace clustering strategy. In both charts, the red line represents the value of the chosen time-dependent or time-independent variation feature. When the red line undergoes minor variations, we assume concept drifts do not occur, while large variations may show concept drifts. The rolling average (green line) is calculated based on a fixed-length sequence of feature values aiming to smooth the data and avoid false positive detections. The blue zones represent the minimum and maximum limits for a feature value change not to be considered significant, i.e., the tolerance boundary (cf. *lower_toler* and *upper_toler* variables in Algorithm 1). The yellow dashed horizontal lines mark the points where concept drifts were detected, while the green dashed horizontal lines mark the points where an actual concept drift exists.

A concept drift is detected when the feature value lies outside the tolerance boundary (i.e., outside the blue zone). The tolerance boundary has a dynamic behavior (increasing and decreasing) based on the variation presented by the rolling average and its standard deviation. For a constant green line, the blue zone is very narrow. Thus, when there are variations in the green line, the blue

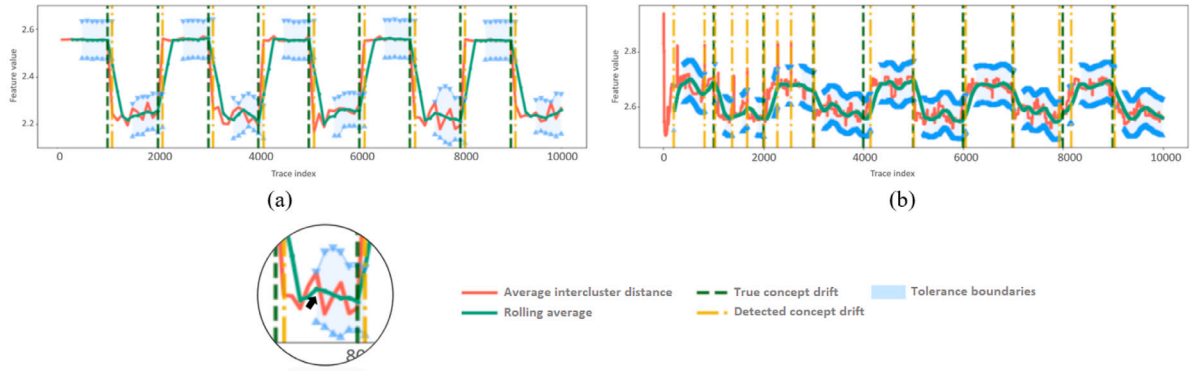


Fig. 8. Example results of running the concept drift detection method for the loan application event log with 10,000 traces considering the change pattern *cb*: (a) using the batch trace clustering strategy; (b) using the stream trace clustering strategy. With the batch trace clustering strategy, there are fewer feature values (one value for each trace window), leading to an overall stability, while with the stream trace clustering strategy, there are more feature values (one for each trace), which tends to cause noise. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

zone needs to be properly relaxed so that small variations in the feature values are not wrongly pointed out as indicative of concept drifts. For example, in Fig. 8(a), near trace 7500, a false concept drift could have been wrongly detected. Regarding effectiveness in detecting concept drifts, all true concept drifts were correctly detected when using the batch trace clustering strategy with exactly one trace delay window (Fig. 8(a)). Likewise, when using the stream trace clustering strategy (Fig. 8(b)), all true concept drifts were correctly detected, although some false positives were detected at the beginning of the trace stream, between trace indexes 0 and 3000, and one at trace index 7900. This is due to the initial tuning phase of the stream clustering method not having been fed enough data yet to become stable.

4.2.3. Effect of parameters

The choice of values for each parameter associated with our approach affects the results in different ways. Sensitivity levels set for both the clustering method and concept drift detection method are directly linked to process concept drift severity. With higher levels of sensitivity, only more severe concept drifts can be detected, while at lower levels of sensitivity, less severe concept drifts can be detected. We summarize as follows our findings on the impact of the values for the parameters on our approach:

- *ws*: Larger windows tend to produce more stable results, as smaller windows may not be able to fully capture the behavior of the process. However, larger windows require more traces to arrive to be completed and allow a batch method iteration to run, possibly delaying detection of concept drift. Window size is a parameter that affects both batch trace clustering and batch non-clustering strategies. Both strategies experience a performance degradation when the window size is set to incorporate multiple drifts. To study drifts in a real-world (unsupervised) environment, it is recommended to begin the exploration with small window sizes.
- *k*, *m*, and *km*: Selecting the number of clusters is a well-known problem in data mining with several ways to solve it [74]; however, its resolution in non-stationary, stream contexts is a challenge. Although high values tend to stabilize the behavior of the feature curves in our approach, that might affect the approach sensitivity leading to detection of more false negatives for concept drifts since increasing the number of clusters allows for a more detailed partitioning of the vector space. On the one hand, this action enables the discovery of more precise local patterns. On the other hand, it may lead to pointing rare traces (outliers) as concept drifts.
- *h* and *r*: Both parameters together produce a similar effect to the *ws* parameter. Low values selected for these two parameters lead to less stability in the feature curves, as they are related to cluster models very sensitive to changes in data distribution. On the other hand, high values for *h* (time horizon) can lead to false negatives, as microclusters tend to remain static when observed from a long time horizon. The same effect was observed for *r* (size of microcluster's radius), although with less impact. Smaller radii imply that microclusters are able to represent more specific regions of the data distribution, while larger radii tend to have broader representations, as each microcluster is able to cover larger regions of the data distribution.
- *rw*: As *rw* controls how smooth the feature series should be, it serves to smooth out more unstable segments and reduce false positives. Thus, low values can lead to more false positives, while high values can lead to more false negatives. In the tests conducted in our experiments, lower values resulted in more accurate detections, indicating that the approach exhibits a bias towards false negatives rather than false positives.
- *mb*: If *mb* is set to close to 0, the tolerance boundary is smaller, which leads to false positive detections. On the other hand, if set to very high values, correct concept drifts may be ignored due to too wide limits of the tolerance boundary, leading to false negatives.
- *θ*: *θ* usually affects segments with greater variation, as more stable segments are dealt by *mb*. For low *θ* values, a narrower tolerance boundary is considered and hence more false positives may occur. On the other hand, for very high *θ* values, event abrupt and significant changes may lie within the tolerance boundary, leading to false negatives.

Table 6

Loan application — effectiveness of the concept drift detection method; best results in terms of F-score achieved for all change patterns, for the three concept drift detection strategies, considering the parameter combinations optimized for each case; the best results for each change pattern are highlighted in bold.

Change pattern	Batch trace clustering				Stream trace clustering				Batch non-clustering			
	Activity			Trans	Activity			Trans	Activity			Trans
	Occur	Freq	Order	Occur	Occur	Freq	Order	Occur	Occur	Freq	Order	Occur
cm	0.95	0.95	0.75	0.81	0.64	0.52	0.42	0.53	0.74	0.75	0.74	0.71
pm	–	–	0.82	0.96	–	–	0.42	0.56	–	–	0.78	0.78
re	1.00	0.41	0.72	0.77	0.52	0.57	0.30	0.51	0.68	0.78	0.63	0.74
rp	0.97	0.97	0.74	0.95	0.74	0.52	0.65	0.57	0.75	0.84	0.75	0.74
cp	–	0.93	0.66	0.93	–	0.50	0.39	0.52	–	0.74	0.73	0.77
sw	–	–	0.63	0.96	–	–	0.37	0.60	–	–	0.68	0.78
cd	–	–	0.66	0.81	–	–	0.57	0.53	–	–	0.62	0.71
pl	–	–	0.63	0.98	–	–	0.47	0.48	–	–	0.73	0.74
cf	1.00	1.00	0.71	0.98	0.36	0.61	0.30	0.45	0.64	0.77	0.67	0.68
cb	0.98	0.98	0.76	0.79	0.62	0.61	0.38	0.56	0.71	0.68	0.71	0.70
lp	–	1.00	0.60	0.72	–	0.51	0.62	0.47	–	0.74	0.68	0.72
fr	0.18	0.18	0.75	0.30	0.36	0.59	0.41	0.46	0.77	0.79	0.74	0.77
IRO	0.98	0.88	0.70	0.98	0.46	0.56	0.39	0.51	0.79	0.71	0.79	0.72
IOR	0.95	0.95	0.71	0.96	0.59	0.51	0.57	0.49	0.70	0.85	0.69	0.73
RIO	–	0.97	0.71	0.95	–	0.60	0.33	0.54	–	0.77	0.75	0.80
ROI	0.97	0.96	0.78	0.96	0.88	0.56	0.62	0.54	0.81	0.79	0.81	0.71
OIR	0.97	0.98	0.75	0.98	0.71	0.58	0.47	0.58	0.72	0.69	0.70	0.71
ORI	–	1.00	0.78	0.95	–	0.58	0.44	0.47	–	0.75	0.72	0.74
Average	0.90	0.87	0.71	0.87	0.59	0.56	0.45	0.52	0.73	0.76	0.72	0.74
Stand dev	0.24	0.24	0.06	0.17	0.16	0.04	0.11	0.04	0.05	0.05	0.05	0.03
Minimum	0.18	0.18	0.60	0.30	0.36	0.50	0.30	0.45	0.64	0.68	0.62	0.68
Maximum	1.00	1.00	0.82	0.98	0.88	0.61	0.65	0.6	0.81	0.85	0.81	0.8
#res=1.00	2	3	0	0	0	0	0	0	0	0	0	0

4.2.4. Concept drift detection results

The experiments to evaluate our concept drift detection method were performed considering all combinations among: *features* (cf. Table 3), *trace clustering parameters* (cf. Table 5(a)), and *concept drift detection parameters* (cf. Table 5(b)). These parameter combinations were in turn applied to the three event logs (with 5000, 7500, and 10,000 cases) combined with the 18 change patterns (cf. Table 4). Concept drift detection results were evaluated using *F-Score* to capture the trade-off between precision and recall.

Table 6 summarizes the effectiveness results, via F-Score, of applying our concept drift detection method to all variants of the loan application process. For each change pattern, we present the average values among the three event logs of sizes 5000, 7500 and 10,000 cases, so that the results of our method can be considered independent of the event log size. Only the best results are listed for each trace profile and for each concept drift detection strategy, considering only the best combination of parameters optimized for each case. To select the best parameter combinations, a ranking considering a cost–benefit ratio in terms of which best solves the largest set of change patterns for each trace profile and each concept drift detection strategy was used. The best combinations of parameters resulting from the tests for all parameter variations are shown in Tables 7a, 7b, and 7c. In Table 6, the highest F-score values for each change pattern are highlighted in bold. Change patterns not supported by a given trace profile were disregarded.

According to Table 6, regarding the different clustering strategies, the best results were achieved with the batch trace clustering strategy, which was able to outperform both the stream trace clustering and batch non-clustering strategies for all change patterns regardless of the trace profile used, with the exception of one case (i.e., change pattern *fr*). Moreover, for the three clustering strategies, the *activity-order* and *transition-occurrence* trace profiles were the only ones capable of representing all 12 basic change patterns, as expected according to Table 4. Specifically for the batch trace clustering strategy, *transition-occurrence* presented better results than *activity-order*, except for change pattern *fr*. This could lead to the conclusion that *transition-occurrence* is the best trace profile. However, *activity-occurrence* and *activity-frequency* led to better results for the vast majority of change patterns they support. For example, for change pattern *lp*, which consists of including a loop structure that increases the frequency of a specific activity within a trace, we obtained better results through *activity-frequency*, which handles this behavior more adequately. As for the six composite change patterns (*RIO*, *ROI*, *ORI*, *OIR*, *IRO*, and *IOR*), results similar to the basic change patterns were achieved. These results show that more complex concept drifts are not more difficult to be detected by this strategy.

Since Table 6 lists only the best results overall for each trace profile for each concept drift detection strategy, good results achieved for only some specific change patterns in particular were left out of this summary. For example, overall results in Table 6 shows that change pattern *fr* is the most difficult problem to solve by our approach for all trace profiles for all concept drift detection strategies under evaluation. In fact, this change pattern has a slight impact on the process behavior as it only changes the frequency of two specific activities associated with alternative paths in the process models (related to a XOR gateway), which have only a small influence in the overall clustering situation. We verified our trace clustering-based method is not very sensitive to this type of change. However, some particular parameter combinations showed better results specifically for *fr*, while showing poor results for other change patterns. For instance, the approach is able to achieve F-score = 0.91 for *fr* when using the following parameter values: *trace clustering strategy* = *batch*, *trace profile* = *activity-order*, *feature* = *avg.dist.between.centroids.positions*, *ws* = 150, *k* = 2.0,

Table 7

Loan application — the best combinations of parameters resulting from the tests for all parameter variations for each trace profile for concept drift detection.

(a) Best combination of parameter values for the batch trace clustering strategy							
Trace profile	Feature	Trace clustering		Concept drift detection			
		ws	k	rw	θ	mb	
activity-occurrence	count_non_zero_MSE	125	2	3	1.0		0.3%
activity-frequency	count_non_zero_MSE	125	2	3	1.0		0.3%
activity-order	total_MSE	200	2	3	2.0		1.0%
transition-occurrence	count_non_zero_MSE	125	2	3	2.0		0.5%

(b) Best combination of parameter values for the stream trace clustering strategy								
Trace profile	Feature	Trace clustering				Concept drift detection		
		m	h	t	km	rw	θ	mb
activity-occurrence	avg_macro_intercluster_distance	50	200	2	3	200	1.5	1.0%
activity-frequency	count_non_zero_MSE_macro	50	200	2	3	200	3.5	3.0%
activity-order	avg_macro_centroids_std	50	200	2	3	200	2.5	1.0%
transition-occurrence	std_micro_intercluster_distance	50	200	2	3	200	3.5	3.0%

(c) Best combination of parameter values for the batch non-clustering strategy					
Trace profile	Feature	ws	Concept drift detection		
			rw	θ	mb
activity-occurrence	avg_interpoints_distance	200	3	1.25	0.3%
activity-frequency	std_interpoints_distance	200	3	1.25	0.7%
activity-order	avg_interpoints_distance	200	3	1.25	1.0%
transition-occurrence	std_interpoints_distance	200	3	1.25	0.7%

$rw = 3$, $\theta = 2$, and $mb = 0.7\%$. This phenomenon also occurs for the other change patterns besides fr , i.e., each change pattern may individually present better F-score results than those presented in Table 6 when taking certain parameter combinations.

Table 8 lists the effectiveness results considering the best combination of parameter values for each change pattern individually. Thus, instead of presenting the best overall results for a given trace profile and concept drift detection strategy (as done in Table 6), the best individual result for each particular change pattern is presented, even if different combinations of parameter values were required. For example, for change pattern fr , for *activity-order* and batch trace clustering strategy, the best of all achieved F-Score is listed, i.e., 0.91, as mentioned in the previous paragraph. The *activity-order* trace profile is the one for which the concept drift detection method achieved the best results when the parameter value selection is optimized for each change pattern individually. The complexity in terms of both the number of dimensions and the variance in the vector space model, inherent to *activity-order* and *transition-occurrence*, increases the sensitivity of our approach regarding the adequacy of parameter values. Thus, it might be worth investing in the search for the ideal parameters, specially for the *activity-order* trace profile, which presented the best individual results. Comparing the results of Table 8 with the results of Table 6, in terms of average F-Score for the *activity-order* trace profile, the batch non-clustering strategy improved by an average of 13%, while the batch trace clustering strategy improved in 37% and, finally, the stream trace clustering strategy improved in 93%.

We compared our results with the concept drift detection methods presented in the literature [11,15,52] that were tested on the same set of event logs, and for which the authors have reported the F-score as an evaluation measure. All baselines methods involved in such comparison were implemented with batch trace strategies. When considering the best parameter setting for each change pattern (cf. Table 8), our approach implemented also with batch trace strategy, as well as with a clustering strategy, produces competitive results. In particular, the work presented by Maaradji et al. [15] is the one that most resembles ours in terms of the results obtained. Our approach implemented with batch trace clustering strategy produces results equal to or better than Maaradji et al. [15]'s work for 12 out of 18 change patterns. For both the Maaradji et al. [15]'s work and our batch trace clustering strategy, the average F-Score was 0.97; however, we achieved the best minimum F-Score with 0.80 (for change pattern pl) versus 0.75 (for change pattern fr). As far as our stream trace clustering strategy concerns, while it does not outperform the batch Maaradji et al. [15]'s work, it achieves near-Maaradji et al. [15]'s results for 11 out 18 change patterns, i.e., with differences of up to 10%. This is a relevant performance as this strategy works in a context of stream of traces, which is inherently more complex than the context of batch of traces taking into account the decision-making needs of the method in question. Finally, the batch non-clustering strategy achieves near-Maaradji et al. [15]'s results for only 2 out 18 change patterns, also only considering differences of up to 10%, which shows trace clustering indeed aids to detect concept drifts.

4.2.5. Concept drift localization results

To isolate the evaluation of our concept drift localization method, we assume that the concept drift detection method worked perfectly. We followed this procedure as the goal of this part of the experiment was to evaluate the performance of the localization

Table 8

Loan application — effectiveness of the concept drift detection method; best results in terms of F-score achieved for the *activity-order* trace profile considering the parameter combination optimized specifically for each change pattern and for each concept drift detection strategy, and comparison with the baselines results; the best results for each change pattern are highlighted in bold.

Change pattern	Our approach			Baselines		
	Batch trace clustering	Stream trace clustering	Batch non-clustering	Batch		
	activity-order	activity-order	activity-order	[15]	[11]	[52]
cm	1.00	0.98	0.80	1.00	0.97	0.72
pm	0.96	0.87	0.89	1.00	0.98	1.00
re	1.00	0.90	0.79	1.00	0.90	1.00
rp	1.00	0.96	0.79	0.96	0.97	1.00
cp	1.00	0.78	0.86	1.00	0.98	1.00
sw	0.96	0.82	0.84	1.00	1.00	1.00
cd	0.83	0.76	0.73	0.88	0.95	0.50
pl	0.80	0.67	0.80	1.00	0.95	0.95
cf	1.00	0.96	0.83	0.98	0.98	1.00
cb	1.00	0.96	0.87	0.92	0.97	1.00
lp	0.97	0.91	0.75	1.00	0.76	1.00
fr	0.91	0.68	0.79	0.75	0.98	0.88
IRO	1.00	0.96	0.79	1.00	0.94	–
IOR	1.00	0.86	0.77	1.00	0.96	–
RIO	0.96	0.64	0.81	0.98	0.97	–
ROI	1.00	0.96	0.82	1.00	1.00	–
OIR	1.00	0.92	0.85	0.97	0.73	–
ORI	1.00	1.00	0.78	1.00	0.98	–
Average	0.97	0.87	0.81	0.97	0.94	0.92
St dev	0.06	0.11	0.04	0.06	0.05	0.11
Min	0.80	0.64	0.73	0.75	0.73	0.50
Max	1.00	1.00	0.89	1.00	1.00	1.00
#res=1.00	11	1	0	11	2	8

method by itself, by analyzing its ability to provide meaningful information about all real concept drifts once they have been previously correctly detected.

Fig. 9 shows a visual example (via a grayscale heatmap) of how the detected concept drifts are localized following the MSE variation as presented in Section 3.4. The MSE variation is calculated for each trace window where a concept drift was detected for the change pattern *cb*, for each clustering strategy. In the heatmap, darker cells express higher MSE values, while lighter ones express lower values. The *y*-axis represents the trace indices referring to the event log, while the *x*-axis lists the activities of the loan application process. The dashed horizontal lines show when the true concept drifts occur. The higher the MSE value for each of the 12 activities, observable by the grayscale gradation of the respective cell, the greater the chance of the concept drift has occurred in that activity. Thus, the grayscale gradation shows the sensitivity of each of the three strategies in associating the occurrence of the concept drift with each activity. There are two activities involved in the concept drift illustrated in this example (*Check if home insurance quote is requested* and *Prepare acceptance pack*), as highlighted in the charts. The charts in Fig. 9 allow us to observe that the MSE indications are correct for all detected concept drifts for the batch trace clustering and batch non-clustering strategies, as well as for most detected concept drifts for the stream trace clustering strategy; i.e., that the two highest MSEs were calculated for the expected activities according to the ground truth. As discussed for Fig. 7 in relation to concept drift detection, the potential differences among the nine pairs of log fragments before and after concept drift introduce variabilities in scenarios for concept drift localization as well. Similarly, these scenario variabilities also explain why activities associated with concept drift are correctly identified at certain points but not at others. Just as with detection, the batch trace clustering strategy exhibits greater robustness in handling these variabilities for localization compared to the stream trace clustering and non-clustering strategies.

Table 9 summarizes the effectiveness results of our concept drift localization method for all variants of the loan application process, via our *ad-hoc* approach. The *trace clustering parameters* listed in Table 5(a) were used here as well. To evaluate our concept drift localization method, the experiments were conducted following the same strategy used to evaluate the concept drift detection method. Only the best results are listed in Table 9 for each trace profile and for each concept drift localization strategy, considering only the best combination of parameters optimized for each case. The best combinations of parameters resulting from the tests for all parameter variations are shown in Table 10.

For each change pattern, we present the average values among the three event logs of sizes 5000, 7500 and 10,000 cases, so that the results of our method can be considered independent of the event log size. As with the results found for concept drift detection, the batch trace clustering strategy emerged with the best overall results, followed closely by the stream trace clustering strategy and the batch non-clustering strategy. Considering the average of the results, for the three strategies (batch trace clustering, stream trace clustering, and batch non-clustering), *activity-frequency* always obtained the best result, also always followed by *transition-occurrence*. *Activity-frequency* delivers very good performance for our concept drift localization method, achieving near-perfect performance for all change patterns it supports, particularly when combined with the batch trace clustering strategy. *Transition-occurrence* also performed well overall, with the advantage of being able to cover all change patterns.

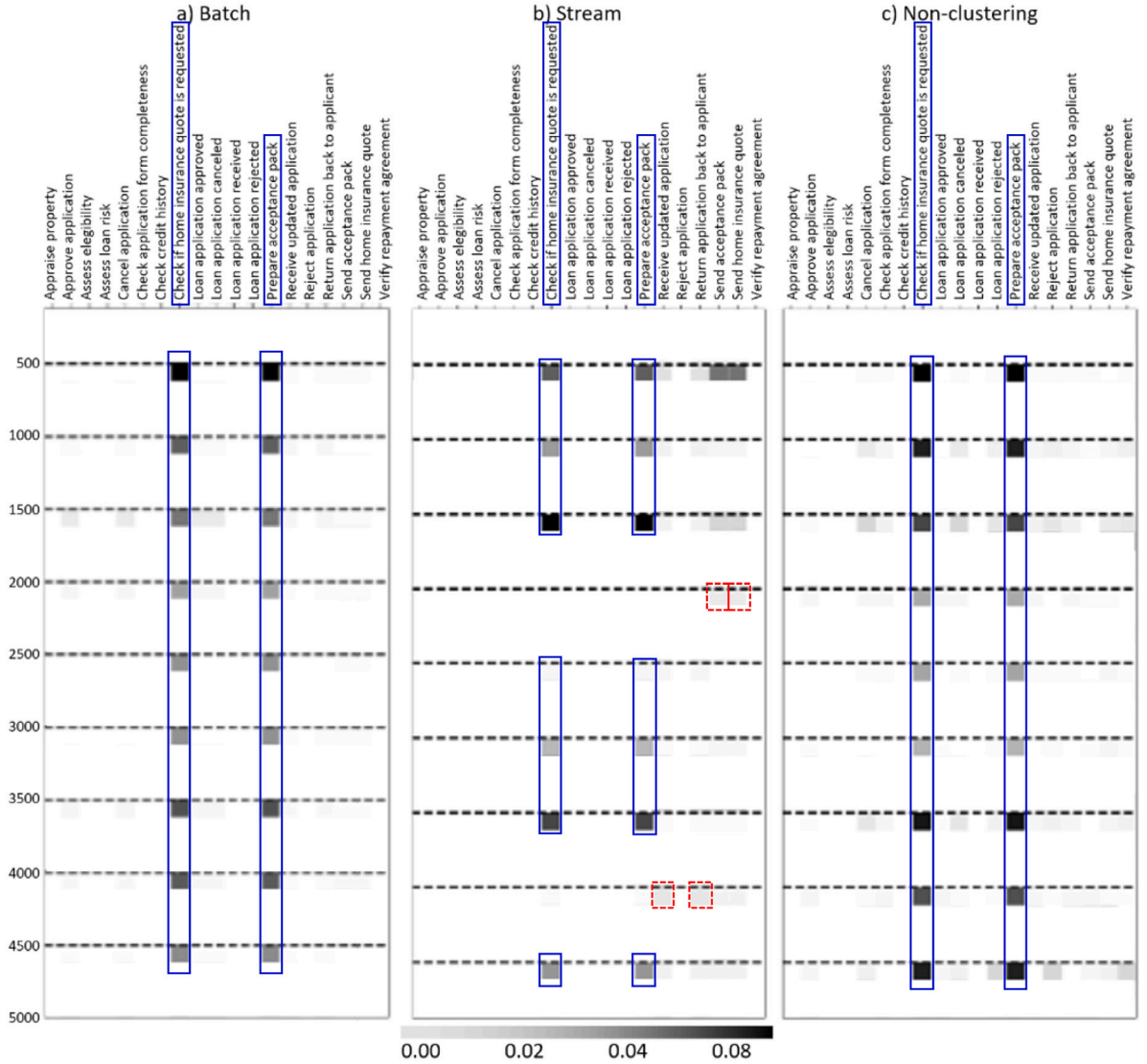


Fig. 9. Loan application — visual example of MSE variations for the three strategies for change pattern *cb* (event log with 5000 traces, *activity-occurrence* trace profile, $ws = 125$, $k = 2$, $h = 200$, $km = 3$, $m = 50$, and $t = 2$). Blue solid markings highlight activities correctly identified as involved in concept drifts, while red dashed markings highlight activities incorrectly identified as involved in concept drifts. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Still in Table 9, one can see that the concept drift localization method achieved good results for change pattern *fr* like the other change patterns for the same parameter combinations. This is a different result than shown in Table 6 for concept drift detection. We see that the concept drift localization method obtained good results for *fr* despite its inherent characteristics already discussed for concept drift detection, because even small variations (in the MSE calculation) can be significant for the localization method, which does not apply for the detection method.

By checking the best combinations of trace clustering parameters selected for the concept drift localization for the batch trace clustering strategy, one can see that $k = 2$ is always present, while $ws = 125$ is present 75% of the time, exactly the same as the concept drift detection experiments. This finding reduces the search space for better parameter combinations when dealing with processes with characteristics similar to the loan application process (e.g., structured processes, with short loops, and two or more final branches).

4.3. Process with high trace variability and noise

The event logs presented by Ostovar et al. [19] are based on a more complex process model than the loan application process used as a basis in Section 4.2. This set of event logs is intended to simulate a noisy environment with high trace variability. The

Table 9

Loan application — effectiveness of the concept drift localization method; best results following our ad-hoc approach achieved for all change patterns, for the three concept drift localization strategies; results in which the area under the curve reached 1.0 for all 27 occurrences of concept drift are designated with the \star symbol; results in which the area under the curve reached falls within the range [0.875, 1.0) for all 27 occurrences of concept drift, in average, are designated with the Δ symbol; results in which the area under the curve reached falls within the range [0.0, 0.875) for all 27 occurrences of concept drift, in average, are left unmarked; and the symbol — indicates that the respective representation does not permit evaluating the change pattern.

Change pattern	Batch trace clustering				Stream trace clustering				Batch non-clustering			
	Activity			Trans	Activity			Trans	Activity			Trans
	Occur	Freq	Order		Occur	Freq	Order		Occur	Freq	Order	
cm	Δ	Δ	Δ	Δ	Δ	—	Δ	\star	Δ	Δ	Δ	\star
pm	—	—	\star	Δ	—	—	Δ	Δ	—	—	Δ	Δ
re	\star	Δ	\star	Δ	\star	Δ	\star	Δ	\star	\star	\star	Δ
rp	\star	\star	\star	\star	Δ	Δ	Δ	Δ	\star	\star	\star	\star
cp	—	\star	—	Δ	—	Δ	—	Δ	—	\star	—	Δ
sw	—	—	Δ	Δ	—	—	—	Δ	—	—	—	Δ
cd	—	—	—	Δ	—	—	—	—	—	—	—	Δ
pl	—	—	—	Δ	—	—	—	—	—	—	—	Δ
cf	\star	\star	\star	Δ	Δ	—	Δ	Δ	Δ	\star	\star	Δ
cb	\star	\star	\star	Δ	Δ	—	Δ	Δ	\star	\star	\star	Δ
lp	—	Δ	—	—	—	Δ	—	—	—	\star	—	—
fr	\star	Δ	\star	Δ	Δ	—	Δ	—	Δ	Δ	Δ	Δ
IRO	—	Δ	Δ	—	—	Δ	—	Δ	—	Δ	—	—
IOR	\star	\star	\star	Δ	Δ	Δ	Δ	—	Δ	\star	\star	Δ
RIO	—	Δ	—	\star	—	—	—	Δ	—	Δ	—	\star
ROI	\star	\star	\star	Δ	Δ	Δ	Δ	Δ	\star	\star	\star	Δ
OIR	—	Δ	Δ	Δ	—	Δ	—	Δ	—	Δ	Δ	Δ
ORI	—	Δ	—	\star	—	Δ	—	Δ	—	Δ	Δ	\star
#res = \star	7	6	8	3	1	0	1	1	4	8	6	4
#res = Δ	1	8	4	13	7	9	8	11	4	5	4	12

Table 10

Loan application — the best combinations of parameters resulting from the tests for all parameter variations for each trace profile for concept drift localization.

Trace profile	Batch		Stream				Non-clustering
	<i>ws</i>	<i>k</i>	<i>h</i>	<i>m</i>	<i>t</i>	<i>km</i>	<i>ws</i>
activity-occurrence	125	2	150	50	2	3	150
activity-frequency	175	2	150	50	2	3	125
activity-order	125	2	200	30	2	3	125
transition-occurrence	125	2	200	30	2	3	125

event logs were created considering fragments of the base process model that has 42 activities, one start event and one end event, five alternative branches, six parallel branches, and three loop structures.¹² The trace variability in this set of event logs ranges from 80% to 99% (considering both original and noisy event logs).

Table 11 lists the change patterns applied by Ostovar et al. [19] to create this set of event logs (similar to those shown in Table 4) and shows which patterns can be represented in each of the four trace profiles used in our approach (marked with a \checkmark sign). There are 13 simple change patterns, also categorized as *I*, *R*, and *O*. As for the composite change patterns, in addition to the six basic composite change patterns established by [15,71], there are six nested composite patterns, i.e., overlapping changes where each change is applied to the fragment resulting from the application of a previous change, named herein with the suffix *n* (e.g., RIO for a composite change and RION for a nested composite change). In total, 25 change patterns were used to create this set of event logs. For each change pattern, five event logs were created, with two concept drifts each, one at trace index 900 and another at trace index 1900. For each event log created, concept drifts involve a different number of activities, ranging from one to five activities. Each event log has 3000 cases in total. For each of the five event logs, versions with 2.5% and 5.0% random noise were also created, totaling 15 event logs per change patterns. The procedure for introducing drifts into the event log should generate sudden-type drifts. Occasionally, behaviors resulting from the insertion of noise can influence the drift design. However, the authors of the event logs do not report any control over the generation of drifts other than sudden ones.

Table 12 lists the parameters (both the trace clustering and the concept drift detection parameters) used in the experiments with this set of event logs, arranged by each trace clustering strategy (similar to those shown in Table 5).

4.3.1. Concept drift detection

Despite the high variability of traces present in this set of event logs, our approach showed a stable recurring pattern for the proposed concept drift detection method at different noise levels, as illustrated in Fig. 10. As expected, by benefiting from the

¹² The process model and further details on creating the event logs are provided by Ostovar et al. [19].

Table 11

High trace variability and noise — change patterns present in event log and support by trace profiles of our approach.

Pattern	Description	Category	Activity			Trans
			Occur	Freq	Order	Occur
pm	Move a fragment into/out of parallel branch	I			✓	✓
sm	Move a fragment to between two fragments	I			✓	✓
cm	Move a fragment into/out of conditional branch	I	✓	✓	✓	✓
rp	Substitute a fragment	I	✓	✓	✓	✓
cr	Insert/delete a fragment in/from conditional branch	I	✓	✓	✓	✓
pr	Insert/delete a fragment in/from parallel branch	I	✓	✓	✓	✓
sr	Insert/delete a fragment between two fragments	I	✓	✓	✓	✓
sw	Swap two fragments	I			✓	✓
pl	Make fragments parallel/sequential	R			✓	✓
cf	Make fragments mutually exclusive/sequential	R	✓	✓	✓	✓
cb	Make a fragment skippable/non-skippable	O	✓	✓	✓	✓
lp	Make a fragment loopable/non-loopable	O		✓	✓	✓
fr	Change branching frequency	O	✓	✓	✓	✓

Table 12

High trace variability and noise — parameters used in the experiments.

Batch trace clustering		Stream trace clustering		Batch non-clustering	
Par	Values	Par	Values	Par	Values
(a) trace clustering parameters					
<i>ws</i>	{100, 125, 150, 175, 200, 250}	<i>m</i>	{30, 50}	<i>ws</i>	{100, 125, 150, 175, 200}
<i>k</i>	{2, 3, 6, 8}	<i>h</i>	{150, 200}		
		<i>t</i>	{2, 50}		
		<i>km</i>	{2, 3, 6}		
(b) concept drift detection parameters					
<i>rw</i>	{3, 4, 5, 7}	<i>rw</i>	{150, 200}	<i>rw</i>	{3, 4}
θ	{1.25, 1.5, 1.75, 2.0}	θ	{1.5, 2.5, 3.5}	θ	{1.25, 1.5, 1.75, 2.0}
<i>mb</i>	{0.3%, 0.5%, 0.7%, 1.0%, 2.5%}	<i>mb</i>	{1.0%, 7.0%, 10.0%}	<i>mb</i>	{0.3%, 0.5%, 0.7%, 1.0%}

generalizability inherent to trace clustering, our method is quite robust in all those scenarios, always exhibiting a stable pattern between concept drift scenarios. Also as expected, the stream trace clustering strategy is more sensitive to fluctuations overall, as it processes each new trace as it appear. Nonetheless, the stream trace clustering strategy usually allows detecting concept drifts with a shorter delay in terms of number of traces, as this strategy is not bound to trace windows and does not require waiting for a specific number of new traces. For the executions with the batch trace clustering method, considering the parameters optimized for each event log, the average delay for detection of drifts is 358.4 traces, for both cases, i.e., with and without noise. As for the executions with the stream trace clustering method, the average drift detection delay drops to 91.3 traces in cases without noise and 91.7 traces in cases with noise.

Table 13 summarizes the effectiveness results, via F-Score, of applying our concept drift detection method to all variants of the high trace variability and noise process. For each change pattern, we averaged the results for the 15 associated event logs, considering the five different numbers of activities involved in the concept drift (i.e., from one to five activities) and the three different noise levels (i.e., 0%, 2.5%, and 5%). Only the best results are listed for each trace profile and for each concept drift detection strategy, considering only the best combination of parameters optimized for each case. The selection of the best parameter combinations followed the same procedure applied for the loan application process, as described in Section 4.2.4. The best combinations of parameters resulting from the tests for all parameter variations are shown in Tables 14a, 14c, and 14b. In Table 13, the highest F-score values for each change pattern are highlighted in bold. Change patterns not supported by a given trace profile were disregarded.

According to Table 13, the batch trace clustering strategy was again responsible for achieving the best results, as reported also for the loan application process. For this set of event logs, the batch trace clustering was able to achieve an F-score of 1.0 for most change patterns. In this case, the *activity-occurrence* trace profile achieved the best result overall, with an average F-score of 0.99 for all change patterns. Moreover, the same level of effectiveness was obtained in simple, composite and nested composite change patterns.

Both the stream trace clustering and batch non-clustering strategies did not show the same level of effectiveness, throughout the different trace profiles and change patterns. Nevertheless, the stream trace clustering strategy is, in fact, also suitable for detecting concept drifts in a scenario of high trace variability and noise, as illustrated in Fig. 10, but this requires a more sensitive configuration of parameters. Therefore, as presented for the loan application process (cf. Section 4.2.4), Table 15 lists the effectiveness results considering the best combination of parameter values for each change pattern individually.

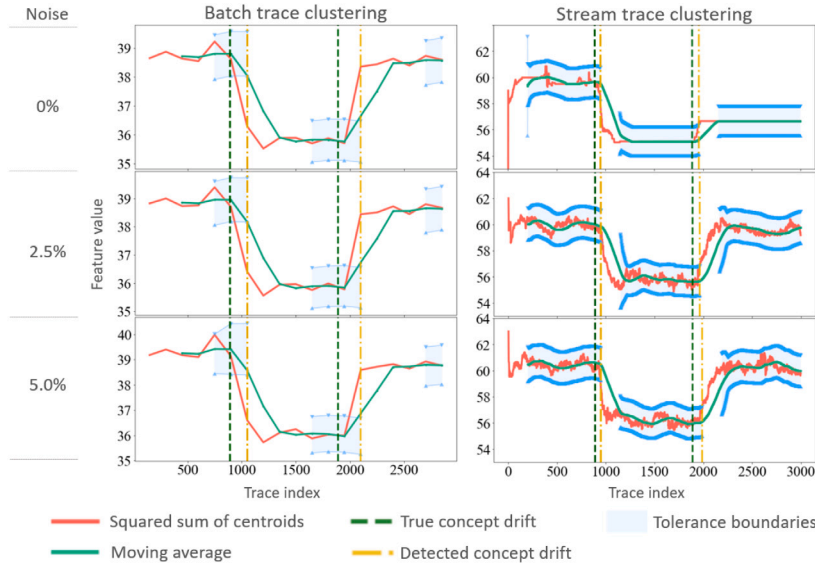


Fig. 10. Illustrative example of running the concept drift detection method for event log *cm-1* with different noise levels. (a) Using $ws = 150$ and $k = 2$ for batch trace clustering strategy, with $rw = 3$, $\theta = 2.5$, and $mb = 2.0\%$ for concept drift detection; (b) Using $m = 30$, $h = 200$, $t = 2$, and $km = 3$ for stream trace clustering strategy, with $rw = 200$, $\theta = 2.5$, and $mb = 2.0\%$ for concept drift detection. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.).

Different trace profiles were responsible for achieving the best effectiveness. For the batch trace clustering strategy, *transition-occurrence* remained the best trace profile. As for the stream trace clustering strategy, *activity-occurrence* was the best trace profile, with the highest average F-Score, although it is not able to deal with all change patterns. Considering only the trace profiles able to find all change patterns, the best one was also *transition-occurrence*. On the other hand, for the batch non-clustering strategy, *activity-order* was the best. Comparing the results of Table 15 with the results of Table 13, in terms of average F-Score, the batch non-clustering strategy improved by an average of 13% for the *activity-order* trace profile, while the stream trace clustering strategy improved in 43% and 44% for *transition-occurrence* and *activity-occurrence*, respectively. Finally, there was no improvement for the batch trace clustering strategy, as the best results had been achieved yet. Although the results were not as good for the stream trace clustering and batch non-clustering strategies as they were for the loan application process, given the greater complexity associated with the process used in this second experiment, the results were better for the batch trace clustering strategy. The low results presented by the batch non-clustering strategy show that, especially for more complex processes, the models provided by trace clustering were able to deliver better results by summarizing the key aspects of the process behavior.

4.3.2. Concept drift localization

We evaluated the concept drift localization method for the high trace variability and noise process following a procedure similar to that applied to the loan application process. We also assumed the concept drift detection method worked perfectly.

Fig. 11 shows a visual example of how the detected concept drifts are localized following the MSE variation similarly to Fig. 9. As for this set of event logs there was no ground truth related to the concept drift localization task, we applied and analyzed our concept drift localization method in an unsupervised manner. The results produced by the concept drift localization method were validated through a visual inspection of the pairs of process models discovered¹³ from the event log stretches before and after the concept drift occurrence. Figs. 11 and 12 illustrate how unsupervised analysis can be conducted.

In Fig. 11, for both batch-based strategies (trace clustering and non-clustering), activities *n* and *p* have the highest MSEs when comparing the process before and after the indices associated with the concept drifts. However, the batch non-clustering strategy additionally has a number of activities with MSEs with intermediate values due to the natural difference in the trace distribution from one trace window to another, which might lead to an error in the concept drift localization. In fact, as expected, trace clustering minimizes natural fluctuations in most variables. As for the stream trace clustering strategy, a high MSE can be observed in activities *r* and *v* for the first concept drift detected, which decreases for the second concept drift. Activities *r* and *v* could be eventually considered a false positive, as can be inferred from the observation illustrated in Fig. 12.

Fig. 12 shows the process models discovered for the traces in each cluster, before and after the first concept drift occurred for the event log *cm-1*. Change pattern *cm* refers to moving a fragment into/out of conditional branch (cf. Table 11). One can see per

¹³ The Inductive Miner algorithm was used for process discovery [73].

Table 13

High trace variability and noise — effectiveness of the concept drift detection method; best results in terms of F-score achieved for all change patterns, for the three concept drift detection strategies, considering the parameter combinations optimized for each case; the best results for each change pattern are highlighted in bold.

Change pattern	Batch trace clustering				Stream trace clustering				Batch non-clustering			
	Activity			Trans	Activity			Trans	Activity			Trans
	Occur	Freq	Order		Occur	Freq	Order		Occur	Freq	Order	
pm	–	–	0.86	1.00	–	–	0.17	0.57	–	–	0.28	0.64
sm	–	–	0.83	1.00	–	–	0.15	0.38	–	–	0.28	0.55
cm	1.00	0.93	1.00	0.97	0.63	0.52	0.65	0.48	0.75	0.53	0.61	0.45
rp	1.00	0.97	1.00	0.93	0.63	0.50	0.62	0.54	0.54	0.47	0.61	0.39
cr	1.00	0.93	1.00	0.91	0.50	0.45	0.63	0.63	0.84	0.46	0.73	0.60
pr	1.00	0.80	1.00	1.00	0.25	0.36	0.37	0.62	0.66	0.55	0.74	0.45
sr	1.00	0.77	1.00	0.97	0.19	0.38	0.33	0.51	0.65	0.51	0.70	0.44
sw	–	–	0.98	1.00	–	–	0.15	0.41	–	–	0.40	0.53
pl	–	–	0.74	1.00	–	–	0.18	0.67	–	–	0.39	0.46
cf	1.00	0.91	1.00	1.00	0.70	0.58	0.74	0.56	0.75	0.48	0.54	0.66
cb	1.00	0.82	1.00	0.97	0.37	0.30	0.34	0.42	0.64	0.44	0.47	0.52
lp	–	1.00	0.95	0.84	–	0.55	0.16	0.38	–	0.53	0.35	0.55
fr	0.86	0.97	0.94	0.90	0.22	0.22	0.25	0.36	0.35	0.48	0.43	0.40
IRO	1.00	1.00	1.00	0.97	0.78	0.60	0.76	0.71	0.69	0.46	0.70	0.61
IOR	1.00	1.00	1.00	0.93	0.67	0.67	0.75	0.66	0.57	0.36	0.46	0.61
RIO	1.00	1.00	1.00	0.97	0.66	0.46	0.57	0.69	0.63	0.53	0.63	0.63
ROI	1.00	1.00	1.00	0.97	0.74	0.64	0.71	0.52	0.78	0.54	0.81	0.62
OIR	1.00	1.00	1.00	0.93	0.69	0.61	0.68	0.72	0.67	0.48	0.62	0.56
ORI	1.00	0.93	1.00	0.93	0.52	0.54	0.43	0.53	0.36	0.33	0.44	0.33
IROn	1.00	0.87	1.00	1.00	0.61	0.57	0.59	0.54	0.46	0.53	0.45	0.48
IORn	1.00	0.97	1.00	1.00	0.69	0.67	0.71	0.45	0.55	0.41	0.67	0.51
RIOn	1.00	1.00	1.00	1.00	0.71	0.58	0.65	0.44	0.64	0.51	0.60	0.66
ROIn	1.00	1.00	1.00	1.00	0.74	0.67	0.76	0.48	0.74	0.43	0.59	0.69
OIRn	1.00	0.89	1.00	1.00	0.48	0.46	0.54	0.59	0.82	0.46	0.75	0.53
ORIn	1.00	1.00	1.00	1.00	0.60	0.60	0.64	0.68	0.58	0.46	0.58	0.56
Average	0.99	0.94	0.97	0.97	0.57	0.52	0.50	0.54	0.63	0.47	0.55	0.54
St Dev	0.03	0.07	0.06	0.04	0.18	0.12	0.22	0.11	0.13	0.06	0.15	0.09
Min	0.86	0.77	0.74	0.84	0.19	0.22	0.15	0.36	0.35	0.33	0.28	0.33
Max	1.00	1.00	1.00	1.00	0.78	0.67	0.76	0.72	0.84	0.55	0.81	0.69
#res=1.00	19	9	19	12	0	0	0	0	0	0	0	0

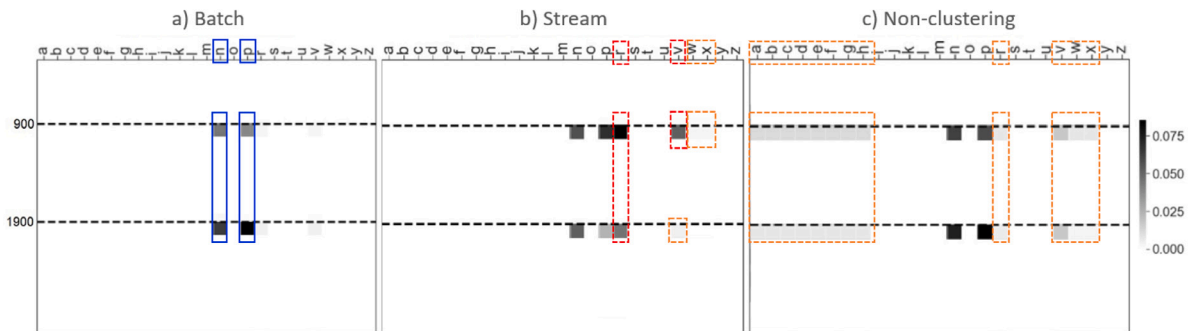


Fig. 11. High trace variability and noise — visual example of MSE variations for the three strategies for change pattern *cm* (event log *cm-1* without noise, activity-occurrence trace profile, $ws = 150$, $k = 3$, $m = 50$, $h = 200$, $t = 2$, and $km = 3$). Considering the unsupervised analysis, blue solid markings highlight activities correctly identified as involved in concept drifts, while red and orange dashed markings highlight activities incorrectly identified as involved in concept drifts. Red color indicates a probable stronger mistake. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 12 that the process model changed in the three clusters in the same part, which is highlighted by the red circle in the figure. Activity *n* (for clusters 0, 1, and 2) and activity *p* (for cluster 0 and 1) were sequential before the concept drift and were moved to a conditional branch. Such analysis confirms the results presented in **Fig. 11** for the batch trace clustering and batch non-clustering strategies. The stream trace clustering strategy in turn also indicated activities *r* and *v*. These false positive indications may be due to the initial adjustment required by our concept drift localization method when working in a trace stream scenario, as the traces are gradually accommodated in the clusters over time.

Table 14

High trace variability and noise — the best combinations of parameters resulting from the tests for all parameter variations for each trace profile for concept drift detection.

(a) Best combination of parameter values for the batch trace clustering strategy								
Trace profile	Feature	Trace clustering		Concept drift detection				
		<i>ws</i>	<i>k</i>	<i>rw</i>	θ	<i>mb</i>		
activity-occurrence	avg_diff_centroids	250	3	3	1.5	1.0%		
activity-frequency	avg_diff_centroids	250	6	3	1.5	1.0%		
activity-order	avg_diff_centroids	250	8	3	1.5	1.0%		
transition-occurrence	avg_diff_centroids	250	8	3	1.5	2.5%		
(b) Best combination of parameter values for the stream trace clustering strategy								
Trace profile	Feature	Trace clustering				Concept drift detection		
		<i>h</i>	<i>m</i>	<i>t</i>	<i>km</i>	<i>rw</i>	θ	<i>mb</i>
activity-occurrence	avg_micro_intercluster_distance	200	50	2	3	200	2.5	7.0%
activity-frequency	count_non_zero_MSE_micro	200	50	2	3	200	2.5	7.0%
activity-order	avg_micro_intercluster_MSE	200	50	2	3	200	2.5	7.0%
transition-occurrence	avg_micro_intercluster_MSE	200	50	2	3	200	2.5	7.0%
(c) Best combination of parameter values for the batch non-clustering strategy								
Trace profile	Feature	<i>ws</i>	Concept drift detection					
			<i>rw</i>	θ	<i>mb</i>			
activity-occurrence	avg_interpoints_max_distance	200	3	1.25	1.0%			
activity-frequency	abs_error	175	3	1.25	0.3%			
activity-order	squared_sum	200	3	1.25	1.0%			
transition-occurrence	avg_points_std	200	3	1.25	1.0%			

5. Experiments with real-world event logs

We applied the approach proposed here in this article to real-world event logs made available by the Business Process Intelligence Challenge (BPIC) event — BPIC 2015 and BPIC 2020. We chose these event logs because they were the subject a previous study [25] related to concept drifts, allowing us to compare our findings with those of other researchers. As there is no ground truth available for these event logs, process models associated with each event log were used as a basis for analyzing the results.

5.1. BPIC 2015

BPIC 2015 comprises a set of five event logs referring to a process for issuing permits to formally allow for construction, demolition, etc. in five Dutch municipalities. The events range from late 2010 to early 2015, with around 50,000 events and 1000 cases for each of the five municipalities. The five municipalities are referred to as *M1*, *M2*, *M3*, *M4*, and *M5*. The process is basically the same for the five municipalities, with minor differences.

There are 356 different activities in these five municipality-related event logs. Of these 356 activities, 35 represent landmark stages of the process, which are identified by the prefix *phase* in their names. Fig. 13 presents a process model discovered by Teinmaa et al. [25] for *M1* considering only the most frequent behavior existing in the event log, resulting in only eight activities of the *phase* type. The process begins with receiving the permit application and then requesting additional information if necessary. Once the application is completely received, there is a phase for obtaining advice known. If the application is ready for decision making, the process jumps directly to the decision making phase. Depending on the decision made, it must be sent for communication, and the process ends.

Over the little more than four years covered by the event logs, changes in procedures, rules, or regulations may have occurred, potentially leading to process concept drifts [25]. To analyze possible changes in process behavior over time, we applied our approach to the five event logs. The experiments were carried out with the three different clustering strategies addressed in our approach, and the parameters were selected empirically. Exploratory experiments were carried out considering the four trace profiles and several different parameter settings. The curves generated from the trace clustering features were visually inspected against each other to choose the clustering parameters. No major differences were found among the trace profiles and thus *activity-occurrence* was selected for further analysis due to its simplicity.

Table 16 presents the parameters used to apply the concept drift detection method. As each event log has a different number of traces, a particular window size was assigned to each of the five event logs as 7% of its total size in number of traces, resulting in the window sizes presented in the table for both the batch trace clustering and batch non-clustering strategies.

Fig. 14 visually shows the results of applying our concept drift detection method to the five municipality-related event logs. Overall, for the five event logs and for batch and stream trace clustering strategies, there is a trend of growth in feature value until

Table 15

High trace variability and noise — effectiveness of the concept drift detection method; best results in terms of F-score achieved considering the parameter combination optimized specifically for each change pattern and for each concept drift detection strategy; the best results for each change pattern are highlighted in bold.

Change pattern	Batch trace clustering	Stream trace clustering		Batch non-clustering
	trans-occur	actv-occur	trans-occur	actv-order
pm	1.00	–	0.83	0.67
sm	1.00	–	0.69	0.51
cm	1.00	0.89	0.66	0.64
rp	1.00	0.86	0.69	0.50
cr	1.00	0.82	0.84	0.74
pr	1.00	0.72	0.83	0.70
sr	1.00	0.58	0.72	0.64
sw	1.00	–	0.87	0.60
pl	1.00	–	0.91	0.54
cf	1.00	0.89	0.72	0.64
cb	0.97	0.62	0.60	0.78
lp	0.94	–	0.53	0.50
fr	0.90	0.40	0.48	0.53
IRO	1.00	1.00	0.95	0.67
IOR	0.97	0.83	0.91	0.72
RIO	1.00	0.89	0.89	0.58
ROI	1.00	1.00	0.79	0.74
OIR	1.00	0.93	0.95	0.68
ORI	1.00	0.71	0.79	0.80
IROn	1.00	0.81	0.79	0.54
IORn	1.00	0.91	0.64	0.62
RIOn	1.00	0.93	0.67	0.50
ROIIn	1.00	0.98	0.71	0.68
OIRn	1.00	0.83	0.85	0.59
ORIn	1.00	0.88	0.91	0.50
Average	0.99	0.82	0.77	0.62
St Dev	0.02	0.15	0.13	0.09
Min	0.90	0.40	0.48	0.50
Max	1.00	1.00	0.95	0.80
#res=1.00	21	2	0	0

Table 16

BPIC 2015 — parameter values used to apply the concept drift detection method.

Clustering strategy	Trace profile	Feature	Trace clustering					Concept drift detection			
			<i>ws</i>	<i>k</i>	<i>h</i>	<i>m</i>	<i>t</i>	<i>km</i>	<i>rw</i>	θ	<i>mb</i>
Batch trace clustering	activity-occurrence	avg_intercluster_distance	83(<i>M</i> 1), 58(<i>M</i> 2), 98(<i>M</i> 3), 73(<i>M</i> 4), 80(<i>M</i> 5)	2	–	–	–	–	100	3.5	10%
Stream trace clustering	activity-occurrence	avg_intercluster_distance	–	–	150	20	2	3	100	3.5	10%
Batch non-clustering	activity-occurrence	avg_intercluster_distance	83(<i>M</i> 1), 58(<i>M</i> 2), 98(<i>M</i> 3), 73(<i>M</i> 4), 80(<i>M</i> 5)	–	–	–	–	–	100	3.5	10%

2012, followed by a trend of stability after 2012. Both batch and stream trace clustering strategies detected, for all municipalities, a concept drift around late 2012 and early-mid 2013, i.e., when the feature value stops growing and becomes more stable. Moreover, for all these cases, there was always a concept drift detected first by the stream trace clustering strategy and then by the batch trace clustering strategy. Other concept drifts were detected by the batch and stream clustering strategies, but without a clear pattern between both strategies or among the five municipalities. As for the batch non-clustering strategy, it detected concept drifts only before or after that period between late 2012 and early-mid 2013.

According to Fig. 14, process changes do not occur at the same time for the five municipalities, probably because each may have implemented such changes at different dates due to possible flexibility in legislation or regulations that require procedural changes [75]. Anyway, different strategies pointing out concept drifts at close points in an event log may show, with a reasonable level of confidence, a true concept drift at that period of process execution. While there is no definitive answer as to whether or not there are concept drifts in this set of event logs, previous studies have also found possible concept drifts in early-mid 2013 [25].

Fig. 15 shows the results of running our concept drift localization method for the five municipality-related event logs, considering concept drifts detected between the end of 2012 and the end of the first half of 2013, i.e., the first concept drift detected for M1, M3, M4, and M5, and the second concept drift detected for M2. These concept drifts occur mainly in activities related to decision making actions –*Phase Decision Ready*, *Phase Draft Decision Ready*, and *Phase Decision Sent*. Specifically for municipality M1, for which the concept drift was detected early, activity *Phase Case Handled* is also pointed out as an activity likely to be involved in the concept drift.

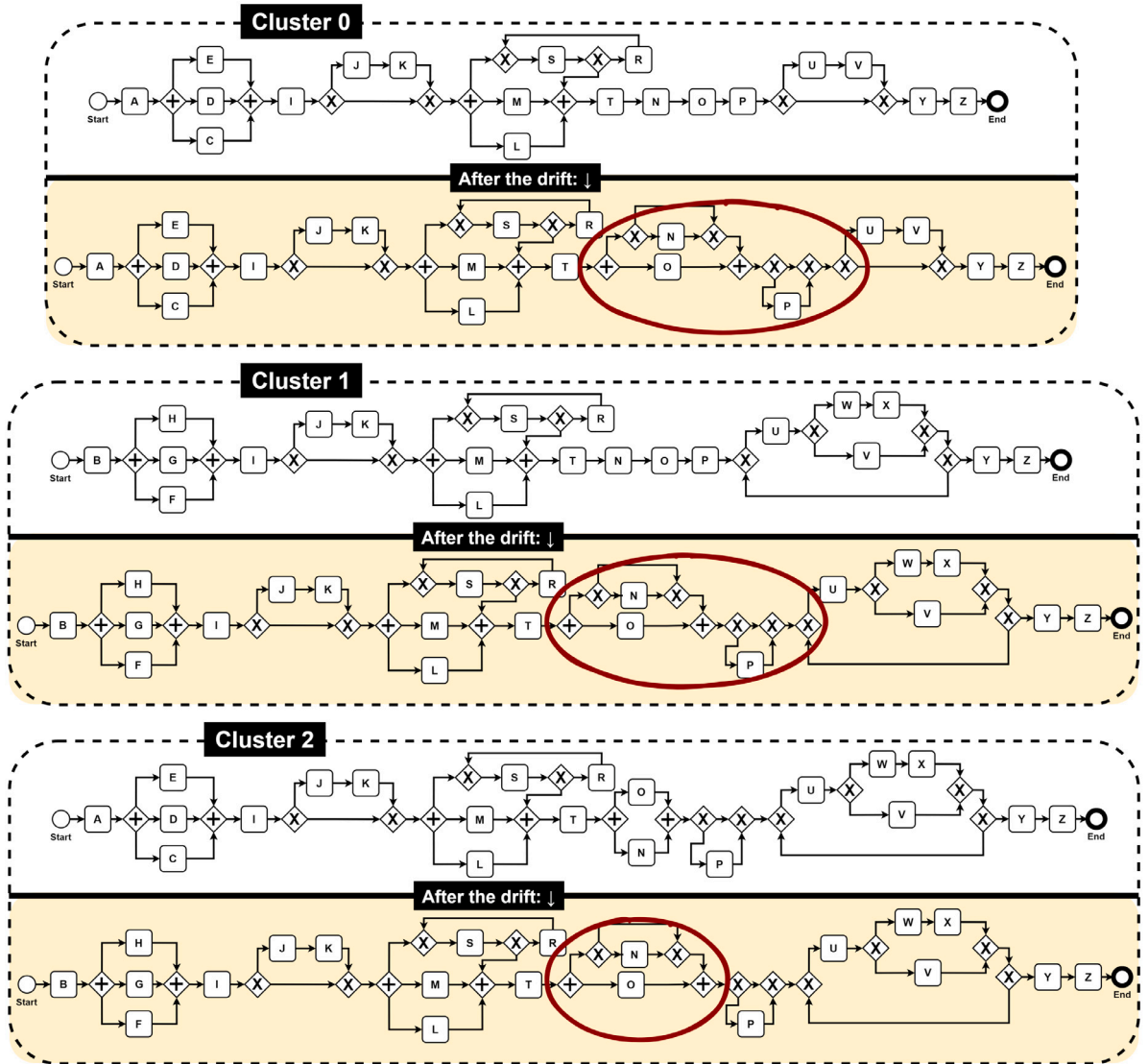


Fig. 12. High trace variability and noise — process models discovered for the traces in each cluster before and after the first concept drift occurred for the event log *cm-1* without noise (batch trace clustering method, *activity-occurrence* trace profile, $ws = 150$, $k = 3$, $m = 50$, $h = 200$, $t = 2$, and $km = 3$).

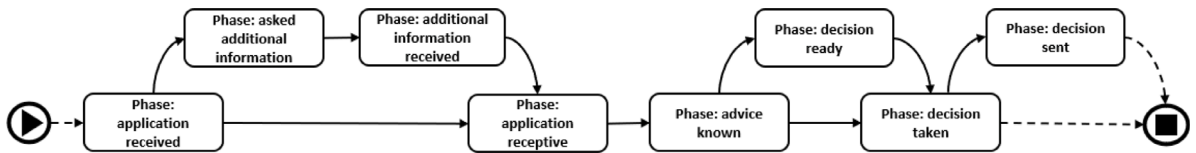


Fig. 13. Discovered process map adapted from Teinemaa et al. [25] for *M1* considering only the most frequent behavior existing in the event log.

Detected potential concept drifts can be validated by process behavior experts to determine which should or should not be considered true concept drifts. Thus, although the experiments reported herein support the hypothesis of the existence of concept drifts in the event logs, these results should be validated by domain experts to obtain definitive answers.

5.2. BPIC 2020

BPIC 2020 comprises a set of event logs referring to a process related to expense claims from several departments at Eindhoven University of Technology (TU/e) [26]. The events range from 2017 to 2018. There are five event logs, with the following sizes:

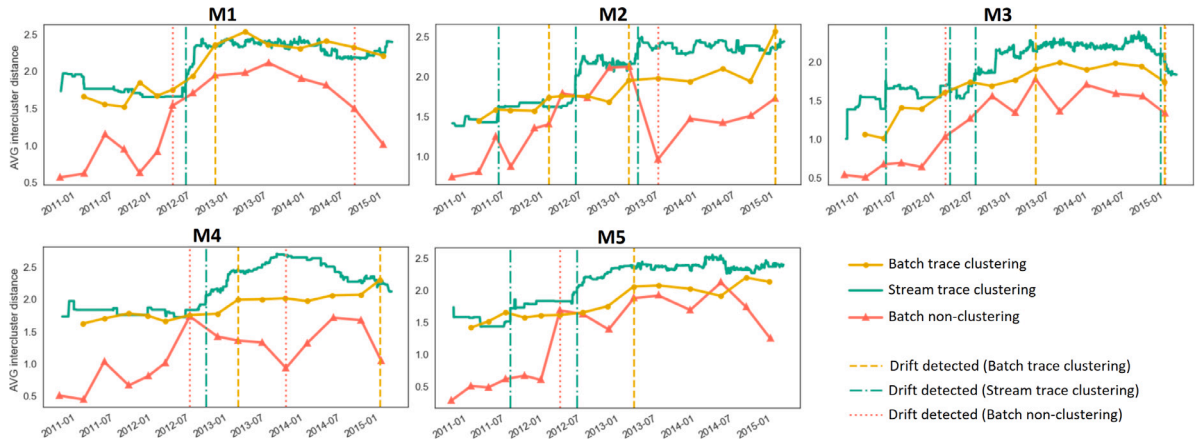


Fig. 14. BPIC 2015 — visual results of applying our concept drift detection method to the five municipality-related event logs, for the *average intercluster distance* feature over time. Vertical lines show concept drifts detected by each strategy.

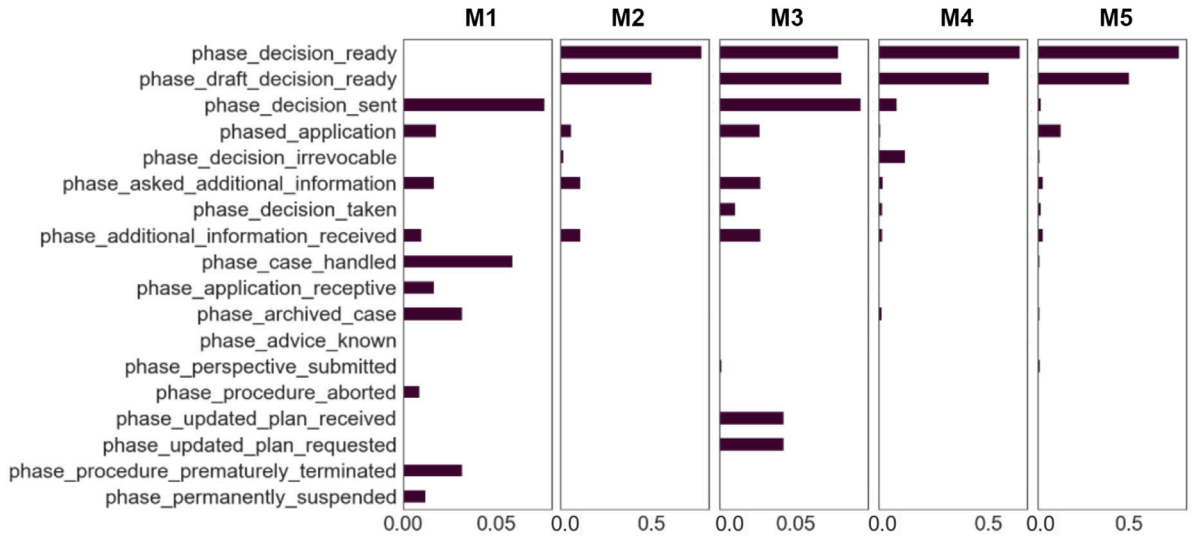


Fig. 15. BPIC 2015 — visual results of applying our concept drift localization method for a concept drifts detected with the batch trace clustering strategy for the five municipality-related event logs. The first concept drift detected for M1, M3, M4, and M5, and the second one detected for M2 are reported. Only activities with $MSE > 0.0$ in at least one event log are shown.

Domestic declarations has around 55,000 events and 10,000 cases; *International declarations*, *Permit*, and *Requests for payment* have around 62,000 events and 6000 cases each one; and *Prepaid travel costs* has around 17,500 events and 2000 cases.

The process underlying this set of event logs is described by van Dongen [26] as the procedures required to request reimbursement of expenses to university staff. Fig. 16 shows a summary model that reflects the process underlying the five event logs, produced by Bano et al. [76]. Under this summarized, integrated process model, the request is initially submitted and then approved by administration, the budget owner, the supervisor (if applicable), and finally the director (if applicable). After all applicable approvals, payment is handled. If the request is not approved by any of the approval instances, the request is rejected and payment not handled.

Similar to what was presented in Section 5.1, Table 17 presents the parameters used to apply the concept drift detection method to the BPIC 2020-related event logs, while Fig. 17 visually shows the results of applying our concept drift detection method to these five event logs.

The feature value curves constructed by the concept drift detection methods show similar behavior for event logs *Domestic Declarations* and *Request for payment*, as well as for event logs *International declarations*, *Permit*, and *Prepaid travel cost*. The results in terms of detected concept drifts for each concept drift detection method can be individually summarized as follows:

- *Stream trace clustering*: This strategy did not detect concept drifts for event logs *International declarations*, *Permit*, and *Request for payment*. For *Domestic declarations* and *Request for payment*, this strategy detected several concept drifts in the initial part of the event log, when the feature value curve is quite unstable. On the other hand, for the same period, the other strategies

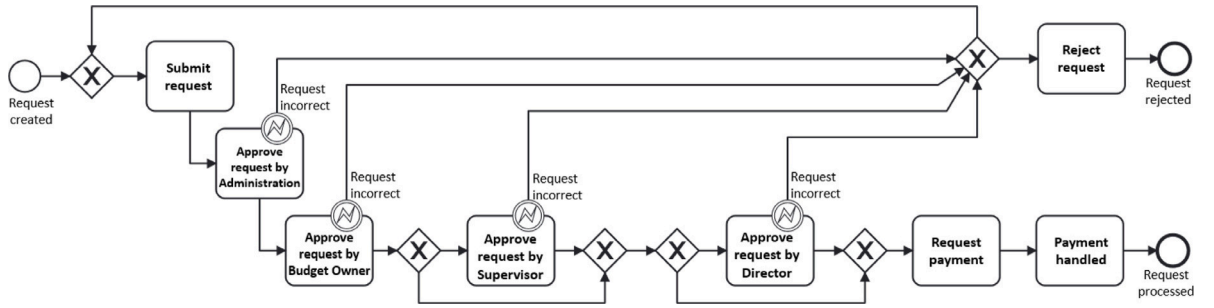


Fig. 16. BPIC 2020 — summary model that reflects the process underlying the five event logs.
Source: Adapted from Bano et al. [76].

Table 17

BPIC 2020 — parameter values used to apply the concept drift detection method.

Clustering strategy	Trace profile	Feature	Trace clustering						Concept drift detection		
			<i>ws</i>	<i>k</i>	<i>h</i>	<i>m</i>	<i>t</i>	<i>km</i>	<i>rw</i>	θ	<i>mb</i>
Batch trace clustering	activity-occurrence	avg_intercluster_distance	225/150 ^a	3	—	—	—	—	3	3.5	7%
Stream trace clustering	activity-occurrence	avg_intercluster_distance	—	—	200	30	2	3	3	3.5	7%
Batch non-clustering	activity-occurrence	avg_intercluster_distance	225/150 ^a	—	—	—	—	—	3	3.5	7%

^a 150 for the *Prepaid travel costs* event log; 225 for all other four event logs.

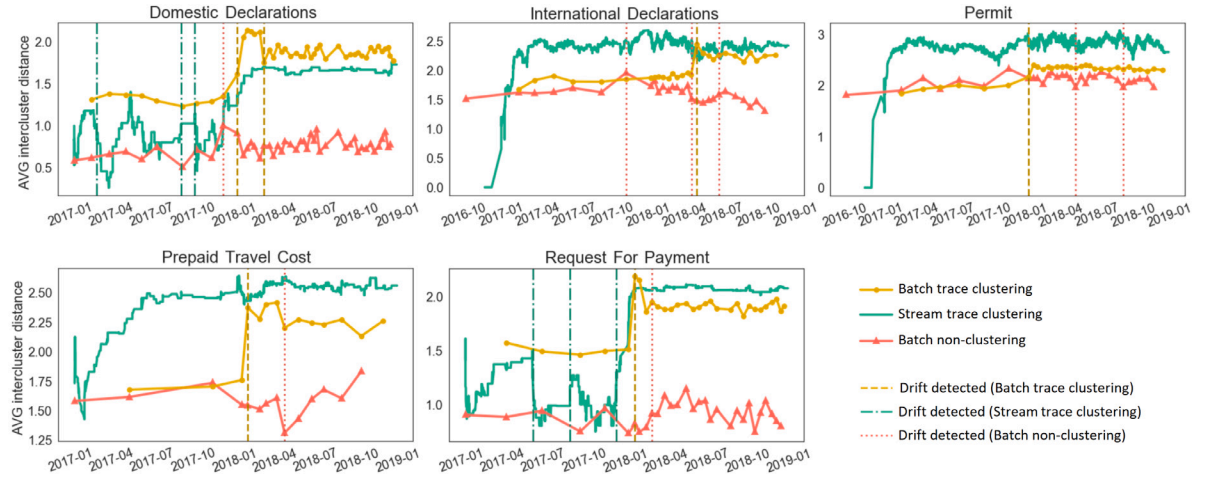


Fig. 17. BPIC 2020 — visual results of applying our concept drift detection method to the five event logs, for the *average intercluster distance* feature over time. Vertical lines show concept drifts detected by each strategy.

presented stable curves and did not detect any concept drift. We hypothesize that this strategy is detecting false positives for this initial period of analysis of the event log since the CluStream algorithm requires an initial time to reach a stable cluster formation.

- *Batch trace clustering*: This strategy detected concept drifts for all event logs, at close moments and with similar feature value curve behavior. A salient difference is that for event log *Domestic declarations* a second concept drift was detected shortly after the first concept drift.
- *Batch non-clustering*: This strategy confirmed the concept drift detected by the *Batch trace clustering* strategy for event log *International declarations*. No clear pattern of concept drifts detected for this strategy was identified for the other event logs.

A similar situation as reported for the BPIC 2015-related event logs (cf. Section 5.1) is found here, i.e., while there is no definitive answer as to whether or not there are concept drifts in this set of event logs, Bano et al. [76] also found evidence of concept drifts in this business process between late 2017 and early 2018. Their hypothesis for the existence of this concept drift is a possible digitalization of this business process that changed to require the digital filling of all cases, which would also explain the increase in the number of cases in this period. Furthermore, the dataset description [26] states that data up to late 2017 was collected from

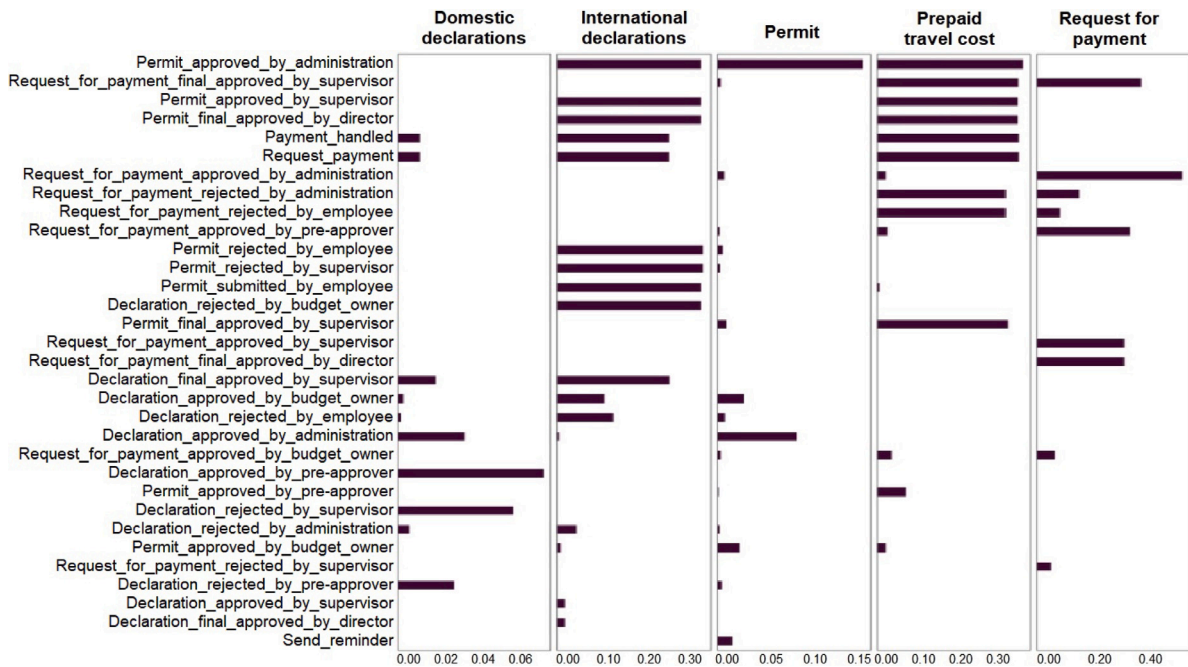


Fig. 18. BPIC 2020 — visual results of applying our concept drift localization method for the first concept drift detected with the batch trace clustering strategy for the five expense claims-related event logs. Only activities with $MSE > 0.0$ in at least one event log are shown.

only two departments, while data from early 2018 onwards was collected from all departments. The addition of new data sources starting in early 2018 may have led to concept drift detection as different departments may work in different ways.

Fig. 18 shows the results of running our concept drift localization method for the five expense claims-related event logs, considering the first concept drift detected with the batch trace clustering strategy. Considering only those activities with the highest MSE for each event log, the activity *Permit_approved_by_administration* is the only one possibly involved in concept drift detected for more than one event log, as it may be involved in the concept drift detected for the event logs *International declarations*, *Permit*, and *Prepaid travel cost*. The activity *Request_for_payment_approved_by_administration* is the most likely to be involved in the concept drift detected for event log *Request for payment*, but only for this event log; while the activity *Declaration_approved_by_pre-approver* is the most likely to be involved in the concept drift detected for event log *Domestic declarations*, but also only for this event log. As for event logs *International declarations* and *Prepaid travel costs*, there are several activities that may be involved in the concept drift detected for each of both event logs. *International declarations* has seven potential activities involved in its detected concept drift, while *Prepaid travel costs* has nine activities; all activities with similar MSE, between 0.30 and 0.35. From these sets of seven and nine activities, three are common for both event logs, namely *Permit_approved_by_administration* (common also with the event log *Permit*), *Permit_approved_by_supervisor*, and *Permit_final_approved_by_director*.

Of the 15 activities potentially involved in the concept drift detected for the five event logs, eight of them (53%) involve activities performed by *administration*, *supervisor*, or *director*, while only three are performed by *employee*. This might show that the concept drifts in these five event logs are related to changes in administrative or supervisory procedures.

By analyzing the content of the event logs, we identified that: (a) event logs *International declarations* and *Prepaid travel cost* have a large number of activities that started or stopped occurring in 2018, leading to a large number of activities with high MSE; (b) event logs *Domestic declarations*, *Permit*, and *Request for payment* have a low number of activities that started or stopped occurring in 2018, leading to a low number of activities with high MSE. Both situations (a) and (b) confirm the content of Fig. 18.

6. Conclusion

We presented in this article an approach to perform concept drift detection and localization in the context of process mining based on trace clustering with the support of batch trace clustering (introduced previously in another paper) and stream trace clustering. We discussed the impact of different parameter values and trace profiles on concept drift detection results. When comparing with baselines for the batch strategy, our approach was able to present equivalent or better results when considering the selection of the best parameter values. Even for event logs with high trace variability and noise and high process complexity, our approach showed good results, although there is no baseline for direct comparison for this case. As for our concept drift localization method, we showed its ability to provide information about the activities potentially involved in the detected concept drift using the same clusters obtained through the concept drift detection method. While the results obtained from real-world event log experiments can

only be confirmed in a supervised manner with the help of domain experts, we showed that our approach can provide insights into possible concept drifts.

The concept drift detection and localization methods presented in this article rely on comparing clustering situations, which hence subject them to the limitations of trace profiles and clustering algorithms. For example, in the experiments conducted, we had to assume to know a priori all the dimensions involved in the event log, i.e., to know both all the possible activities and all the possible activity transitions present in the event log beforehand. While this is a limitation of our approach, knowing at least all possible activities in the event log is feasible on many occasions for real-world processes. Furthermore, the synthetic event logs employed in the experiments were specifically constructed to contain sudden-type drifts. Hence, we contend that our approach demonstrates favorable outcomes for this particular drift characterization. Nonetheless, the effectiveness of our approach for other types of drifts necessitates thorough examination and careful assessment.

Pathways to expand this work include: (i) exploring more trace profiles, such as transition frequency, and combining multiple trace profiles to improve trace clustering, introduced by Appice and Malerba [21]; (ii) exploring density-based stream clustering algorithms, as done by Tavares et al. [39]; (iii) applying noise reduction strategies to improve the concept drift detection and localization methods; (iv) studying the application of methods for automatic selection of parameter values, inspired by strategies that have been proposed for automated machine learning (AutoML) in the context of process mining [77], given the high dependence that the performance of our approach has on parameters; and (v) conducting research to examine and elucidate potential patterns in the relationships between parameter value ranges and characteristics of event logs [78], in order to derive heuristics that can minimize the effort required for parameterization in our approach.

Furthermore, alongside the expansion of our proposed approach, it is pertinent to acknowledge that concept drift analysis remains a significant obstacle in the practical application of data stream mining solutions to real-world problems [79]. Several factors contribute to this challenge, including the scarcity of benchmark datasets and the inadequacy of drift detectors in handling temporal dependencies within the data. These factors are also prevalent in the field of process mining and represent substantial hurdles that must be addressed to facilitate notable advancements in the realm of business process management within online environments.

CRedit authorship contribution statement

Rafael Gaspar de Sousa: Conceptualization, Methodology, Software, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Antonio Carlos Meira Neto:** Validation, Investigation, Data curation, Writing – review & editing. **Marcelo Fantinato:** Writing – review & editing, Visualization, Supervision. **Sarajane Marques Peres:** Conceptualization, Methodology, Writing – review & editing, Visualization, Supervision, Project administration. **Hajo Alexander Reijers:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank Universidade de São Paulo and Amazon Web Services (AWS) for the support of this research in Cloud Manufacturing Services, in the scope of USP-AWS collaboration; the São Paulo Research Foundation – Fapesp-Brazil – process numbers 2017/26487-4 and 2017/26491-1.

Declaration

During the preparation of this work, the authors used ChatGPT based on GPT-3.5 in order to exclusively improve language and readability of some specific parts within the manuscript. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

- [1] U.M. König, A. Linhart, M. Röglinger, Why do business processes deviate? Results from a Delphi study, *Bus. Res.* 12 (2) (2019) 425–453.
- [2] R.P.J.C. Bose, W. van der Aalst, I. Žliobaitė, M. Pechenizkiy, Dealing with concept drifts in process mining, *IEEE Trans. Neural Netw. Learn. Syst.* 25 (1) (2014) 154–171.
- [3] I. Žliobaitė, M. Pechenizkiy, J. Gama, An overview of concept drift applications, in: N. Japkowicz, J. Stefanowski (Eds.), *Big Data Analysis: New Algorithms for a New Society*, Vol. 16, 2016, pp. 91–114.
- [4] G. Krempel, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, J. Stefanowski, Open challenges for data stream mining research, *SIGKDD Explor. Newsl.* 16 (1) (2014) 1–10.
- [5] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, *ACM Comput. Surv.* 46 (4) (2014) 44:1–44:37.
- [6] R.G. de Sousa, S.M. Peres, M. Fantinato, H.A. Reijers, Concept drift detection and localization in process mining: An integrated and efficient approach enabled by trace clustering, in: *36th Annual ACM Symp. on Applied Computing, SAC '21*, Association for Computing Machinery, New York, NY, USA, 2021, pp. 364–373.
- [7] W. van der Aalst, *Process Mining: Data Science in Action*, second ed., 2016.
- [8] W. van der Aalst, Process mining in the large: A tutorial, in: *3rd Eur. Summer Sch. on Bus. Intell.*, 2014, pp. 33–76.

- [9] W.M.P. van der Aalst, Business process management: A comprehensive survey, ISRN Softw. Eng. 2013 (2013).
- [10] P.M. Gonçalves Jr., S.G. de Carvalho Santos, R.S. Barros, D.C. Vieira, A comparative study on concept drift detectors, *Expert Syst. Appl.* 41 (18) (2014) 8144–8156.
- [11] A. Seeliger, T. Nolle, M. Mühlhäuser, Detecting concept drift in processes using graph metrics on process graphs, in: 9th Conf. on Subject-Oriented Bus. Process Manage., 2017.
- [12] A. Burattin, Streaming process mining, in: W.M.P. van der Aalst, J. Carmona (Eds.), *Process Mining Handbook*, Springer Int'l Publishing, Cham, 2022.
- [13] R.P.J.C. Bose, W. van der Aalst, I. Žliobaitė, M. Pechenizkiy, Handling concept drift in process mining, in: 23rd Int. Conf. on Adv. Inf. Syst. Eng., 2011, pp. 391–405.
- [14] J. Carmona, R. Gavalda, Online techniques for dealing with concept drift in process mining, in: 11th Int. Symp. Adv. in Intel. Data Anal., 2012, pp. 90–102.
- [15] A. Maaradj, M. Dumas, M.L. Rosa, A. Ostovar, Fast and accurate business process drift detection, in: 13th Int'l Conf. on Bus. Process Manage., 2015, pp. 406–422.
- [16] C. Zheng, L. Wen, J. Wang, Detecting process concept drifts from event logs, in: Int. Conf. on Coop. Inf. Sys., 2017, pp. 524–542.
- [17] M. Song, C.W. Günther, W. van der Aalst, Trace clustering in process mining, in: 4th Works. on Bus. Process Intell., 2008, pp. 51–62.
- [18] R.J.C. Bose, W. van der Aalst, Context aware trace clustering: Towards improving process mining results, in: SIAM Int. Conf. on Data Mining, 2009, pp. 401–412.
- [19] A. Ostovar, S.J.J. Leemans, M.L. Rosa, Robust drift characterization from event streams of business processes, *ACM Trans. Knowl. Discov. Data* 14 (3) (2020).
- [20] G. Greco, A. Guzzo, L. Pontieri, D. Saccà, Discovering expressive process models by clustering log traces, *IEEE Trans. Knowl. Data Eng.* 18 (2006) 1010–1027.
- [21] A. Appice, D. Malerba, A co-training strategy for multiple view clustering in process mining, *IEEE Trans. Serv. Comput.* 9 (6) (2016) 832–845.
- [22] J. De Weerd, S.K. vanden Broucke, J. Vanthienen, B. Baesens, Leveraging process discovery with trace clustering and text mining for intelligent analysis of incident management processes, in: 2012 IEEE Congress on Evolutionary Computation, 2012, pp. 1–8.
- [23] J. De Weerd, S. vanden, J. Vanthienen, B. Baesens, Active trace clustering for improved process discovery, *IEEE Trans. Knowl. Data Eng.* 25 (12) (2013) 2708–2720.
- [24] B. Hompes, J. Buijs, W. van der Aalst, P. Dixit, J. Buurman, Detecting Changes in Process Behavior Using Comparative Case Clustering, in: *Lecture Notes in Business Information Processing*, 2017, pp. 54–75.
- [25] I. Teinemaa, A. Leontjeva, K.-O. Masing, BPIC 2015: Diagnostics of building permit application process in dutch municipalities, in: *Fifth Int'l Business Process Intelligence Challenge*, BPIC'15, 2015.
- [26] B. van Dongen, BPIC Challenge 2020, 4TU.ResearchData, 2020.
- [27] D.M.V. Sato, S.C. De Freitas, J.P. Barddal, E.E. Scalabrin, A survey on concept drift in process mining, *ACM Comput. Surv.* 54 (9) (2021) <http://dx.doi.org/10.1145/3472752>.
- [28] B. Krawczyk, A. Cano, Online ensemble learning with abstaining classifiers for drifting and noisy data streams, *Appl. Soft Comput.* 68 (C) (2018) 677–692.
- [29] A. Ostovar, A. Maaradj, M.L. Rosa, A.H. ter Hofstede, B. van Dongen, Detecting drift from event streams of unpredictable business processes, in: 35th Int. Conf. on Conceptual Model., 2016.
- [30] A. Ostovar, A. Maaradj, M.L. Rosa, A. ter Hofstede, Characterizing drift from event streams of business processes, in: 29th Int. Conf. on Adv. Inf. Syst. Eng., 2017, pp. 210–228.
- [31] J. Martjushev, R.P.J.C. Bose, W. van der Aalst, Change point detection and dealing with gradual and multi-order dynamics in process mining, in: 14th Int'l Conf. on Perspectives in Bus. Informatics Research, 2015, pp. 161–178.
- [32] P. Weber, B. Bordbar, P. Tino, Real-time detection of process change using process mining, in: *Imperial College Computing Student Works*, 2011, pp. 108–114.
- [33] D. Luengo, M. Sepúlveda, Applying clustering in process mining to find different versions of a business process that changes over time, in: 7th Int'l Works. on Business Process Intelligence, 2012, pp. 153–158.
- [34] R. Accorsi, T. Stocker, Discovering workflow changes with time-based trace clustering, in: 2nd Int. Symp. on Data-Driven Proc. Discov. and Anal., 2012, pp. 154–168.
- [35] M. Kumar, L. Thomas, A. Basava, Capturing the sudden concept drift in process mining, in: Int. Works. on Algor. & Theories for the Anal. of Event Data, 2015, pp. 132–143.
- [36] B.F.A. Hompes, J. Buijs, W. van der Aalst, P. Dixit, J. Buurman, Detecting change in processes using comparative trace clustering, in: 5th Int. Symp. on Data-Driven Proc. Discov. and Anal., 2015, pp. 95–108.
- [37] A. Maaradj, M. Dumas, M.L. Rosa, A. Ostovar, Detecting sudden and gradual drifts in business processes from execution traces, *IEEE Trans. Knowl. Data Eng.* 29 (10) (2017) 2140–2154.
- [38] S.B. Junior, G.M. Tavares, V.G.T.d. Costa, P. Ceravolo, E. Damiani, A framework for human-in-the-loop monitoring of concept-drift detection in event log stream, in: *Companion the Web Conf. 2018*, 2018, pp. 319–326.
- [39] G.M. Tavares, P. Ceravolo, V.G. Turrissi Da Costa, E. Damiani, S. Barbon Junior, Overlapping analytic stages in online process mining, in: 2019 IEEE Int'l Conf. on Services Computing, SCC, 2019, pp. 167–175.
- [40] N. Liu, J. Huang, L. Cui, A framework for online process concept drift detection from event streams, in: IEEE Int. Conf. on Serv. Comput., 2018, pp. 105–112.
- [41] F. Stertz, S. Rinderle-Ma, Process Histories - Detecting and Representing Concept Drifts Based on Event Streams, in: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, pp. 318–335.
- [42] F. Richter, T. Seidl, Looking into the TESSERACT: Time-drifts in event streams using series of evolving rolling averages of completion times, *Inf. Syst.* (2019) 265–282.
- [43] S. Pauwels, An anomaly detection technique for business processes based on extended dynamic Bayesian networks, in: *ACM Symp. on Applied Computing*, 2019, pp. 494–501.
- [44] M. Hassani, Concept drift detection of event streams using an adaptive window, in: *European Council for Modelling and Simulation*, 2019, pp. 230–239.
- [45] A. Kurniati, C. McInerney, K. Zucker, G. Hall, D. Hogg, O. Johnson, A Multi-level Approach for Identifying Process Change in Cancer Pathways, in: *Lecture Notes in Business Information Processing*, 2019, pp. 595–607.
- [46] A. Kurniati, C. McInerney, K. Zucker, G. Hall, D. Hogg, O. Johnson, Using a multi-level process comparison for process change analysis in cancer pathways, *Int. J. Environ. Res. Public Health* (2020) 1–16.
- [47] A. Yeshchenko, C. Ciccio, J. Mendling, A. Polyvyanyy, Comprehensive process drift analysis with the visual drift detection tool, in: *ER Forum and Poster & Demos Session Co-Located with 38th Int'l Conf. on Conceptual Modeling*, 2019, pp. 108–112.
- [48] A. Yeshchenko, C.D. Ciccio, J. Mendling, A. Polyvyanyy, Comprehensive process drift detection with visual analytics, in: 38th Int'l Conf. on Conceptual Modeling, 2019.
- [49] A. Yeshchenko, J. Mendling, C. Di Ciccio, A. Polyvyanyy, VDD: A visual drift detection system for process mining, in: *Tool Demonstration Track Co-Located with the 2nd Int'l Conf. on Process Mining*, 2020, pp. 31–34.

- [50] A. Yeshchenko, C. Di Ciccio, J. Mendling, A. Polyvyanyy, Visual drift detection for sequence data analysis of business processes, *IEEE Trans. Vis. Comput. Graphics* PP (2021).
- [51] T. Brockhoff, M.S. Uysal, W.M. van der Aalst, Time-aware concept drift detection using the earth mover's distance, in: 2020 2nd Int'l Conf. on Process Mining, ICPM, 2020, pp. 33–40.
- [52] L. Lin, L. Wen, L. Lin, J. Pei, H. Yang, LCDD: Detecting business process drifts based on local completeness, *IEEE Trans. Serv. Comput.* (2020).
- [53] A. Impedovo, P. Mignone, C. Loglisci, M. Ceci, Simultaneous process drift detection and characterization with pattern-based change detectors, in: 23rd Int'l Conf. on Discovery Science, Springer Int'l Publishing, Cham, 2020, pp. 451–467.
- [54] J.N. Adams, S.J. van Zelst, L. Quack, K. Hausmann, W.M.P. van der Aalst, T. Rose, A framework for explainable concept drift detection in process mining, in: 19th Int'l Conf. on Business Process Management, Springer-Verlag, Berlin, Heidelberg, 2021, pp. 400–416.
- [55] Y. Lu, Q. Chen, S. Poon, A Robust and Accurate Approach to Detect Process Drifts from Event Streams, in: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2021, pp. 383–399.
- [56] Y. Lu, Q. Chen, S. Poon, Detecting and Understanding Branching Frequency Changes in Process Models, in: *Lecture Notes in Business Information Processing*, 2021, pp. 39–46.
- [57] F. Richter, A. Maldonado, L. Zellner, T. Seidl, OTOSO: Onlin trace ordering for structural overviews, in: 1st Int'l Works. on Streaming Analytics for Process Mining, Springer Int'l Publishing, 2021, pp. 218–229.
- [58] L. Zellner, F. Richter, J. Sontheim, A. Maldonado, T. Seidl, Concept drift detection on streaming data with dynamic outlier aggregation, in: 1st Int'l Works. on Streaming Analytics for Process Mining, Springer Int'l Publishing, 2021, pp. 206–217.
- [59] D.M.V. Sato, J.P. Barddal, E.E. Scalabrin, Interactive process drift detection framework, in: 20th Int'l Conf. on Artificial Intelligence and Soft Computing, Springer Int'l Publishing, Cham, 2021, pp. 192–204.
- [60] L. Yang, S. McClean, M. Donnelly, K. Burke, K. Khan, Process duration modelling and concept drift detection for business process mining, in: *IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation*, 2021, pp. 653–658.
- [61] M. Carnein, H. Trautmann, Optimizing data stream representation: An extensive survey on stream clustering algorithms, *Bus. Inf. Syst. Eng.* 61 (2019) 277–297.
- [62] D. Arthur, S. Vassilvitskii, K-Means++: The advantages of careful seeding, in: 18th Annual ACM-SIAM Symp. on Discrete Algorithms, SODA '07, 2007, pp. 1027–1035.
- [63] J. Silva, E. Faria, R. Barros, E. Hruschka, A. de Carvalho, J. Gama, Data stream clustering: A survey, *ACM Comput. Surv.* 46 (2014).
- [64] C.C. Aggarwal, J. Han, J. Wang, P.S. Yu, A framework for clustering evolving data streams, in: 29th Int'l Conf. on Very Large Data Bases, Vol. 29, VLDB '03, VLDB Endowment, 2003, pp. 81–92.
- [65] C.M.M. Pereira, R.F. de Mello, A comparison of clustering algorithms for data streams, in: *First Int'l Conf. on Integrated Computing Technology*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 59–74.
- [66] M. Spiliopoulou, I. Ntoutsis, Y. Theodoridis, R. Schult, MONIC: Modeling and monitoring cluster transitions, in: 12th ACM SIGKDD Int. Conf. on Knowl. Discovery and Data Mining, 2006, pp. 706–711.
- [67] M. Oliveira, J. Gama, MEC - monitoring clusters' transitions, in: 5th Starting AI Researchers' Symp., 2010, pp. 212–224.
- [68] P.J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, *J. Comput. Appl. Math.* 20 (1987) 53–65.
- [69] D.L. Davies, D.W. Bouldin, A cluster separation measure, *IEEE Trans. Pattern Anal. Mach. Intell. PAMI-1* (2) (1979) 224–227.
- [70] H.W. Kuhn, The Hungarian method for the assignment problem, *Nav. Res. Logist. Q.* 2 (1–2) (1955) 83–97.
- [71] A. Maaradji, M.M. Dumas, M.M.L. Rosa, A.A. Ooctovar, Business Process Drift, 2015, <http://dx.doi.org/10.4121/uuid:aa01a720-4616-43e9-af67-370942019f48>, URL: https://data.4tu.nl/articles/dataset/Business_Process_Drift/12712436.
- [72] M. Dumas, M.L. Rosa, J. Mendling, H.A. Reijers, *Fundamentals of Business Process Management*, 2013.
- [73] S.J.J. Leemans, D. Fahland, W.M.P.v.d. Aalst, Discovering block-structured process models from event logs - a constructive approach, in: 34th Int'l Conf. on Application and Theory of Petri Nets and Concurrency, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 311–329.
- [74] C. Yuan, H. Yang, Research on K-value selection method of K-means clustering algorithm, *J* 2 (2) (2019) 226–235.
- [75] U. van der Ham, Benchmarking of five dutch municipalities with process mining techniques reveals opportunities for improvement, in: *Fifth Int'l Business Process Intelligence Challenge, BPIC'15*, 2015.
- [76] D. Bano, M. Völker, S. Remy, H. Leopold, M. Weske, Multi-perspective analysis of approval processes based on multiple event logs, in: *Tenth Int'l Business Process Intelligence Challenge, BPIC'20*, 2020.
- [77] G.M. Tavares, S. Barbon Junior, E. Damiani, P. Ceravolo, Selecting optimal trace clustering pipelines with meta-learning, in: 11th Brazilian Conf. on Intelligent Systems, Springer Int'l Publishing, Cham, 2022, pp. 150–164.
- [78] G.M. Tavares, S.B. Junior, E. Damiani, Automating process discovery through meta-learning, in: 28th Int'l Conf. on Cooperative Information Systems, Springer Int'l Publishing, Cham, 2022, pp. 205–222.
- [79] S. Wares, J. Isaacs, E. Elyan, Data stream mining: methods and challenges for handling concept drift, *SN Appl. Sci.* 1 (2019).

Rafael Gaspar de Sousa holds a bachelor's degree in Information Systems and a master's in science from the School of Arts, Sciences and Humanities at the University of São Paulo, Brazil. He works as a data scientist at Banco Itaú, Brazil.

Antonio Carlos Meira Neto holds a bachelor's degree in Information Systems and is currently pursuing a master's in science from the School of Arts, Sciences and Humanities at the University of São Paulo, Brazil. He works as a data scientist at Banco Itaú, Brazil.

Marcelo Fantinato is an associate professor of the School of Arts, Sciences and Humanities (EACH) at the University of São Paulo. He is currently a productivity research fellow at CNPq, Brazil. He was a guest researcher at Vrije Universiteit Amsterdam and at Utrecht University, The Netherlands. He has Green Belt certification in the Six Sigma Quality Improvement Program from Motorola. He was the coordinator of the Graduate Program in Information Systems at USP and is the current chair of the graduate committee of EACH-USP. He is a member of the IEEE Technical Committee on Services Computing. He represents the University of São Paulo in the European Research Center for Information Systems. His main research interests are related to business process management, process mining, and social robots.

Sarajane Marques Peres is an associate professor of the School of Arts, Sciences and Humanities at the University of São Paulo (USP). She co-wrote a Data Mining textbook published in Portuguese, worked as the tutor of the PET Information Systems USP group under the Tutorial Education Program of the Ministry of Education, Brazil, and worked as a guest researcher at the Vrije Universiteit Amsterdam and the Utrecht University, The Netherlands. She is member of the researcher board of the Advanced Institute for Artificial Intelligence A12 (Brazil) and C4AI — Center for Artificial Intelligence (USP/IBM/Fapesp). Her main research interests are computational intelligence, data mining and machine learning applied to text mining, process mining, gesture analysis, and human-robot interaction.

Hajo Alexander Reijers is a full professor in the Department of Information and Computing Sciences of Utrecht University, where he holds the chair in Business Process Management and Analytics. He is also a part-time, full professor in the Department of Mathematics and Computer Science of Eindhoven University of Technology, as well as an adjunct professor in the School of Information Systems of Queensland University of Technology. Previously, he headed a research unit within Lexmark and led IT projects as a management consultant for Accenture and Deloitte. The focus of his research is on business process innovation, process analytics, robotic process automation, and enterprise IT. On these and other topics, he published over 200 scientific papers, book chapters, and professional publications. His latest research is concerned with how to let people and computer systems work together gracefully within business processes.