A Framework for Explainable Concept Drift Detection in Process Mining

Jan Niklas Adams[1], Sebastiaan J. van Zelst[2,1], Lara Quack[2], Kathrin Hausmann[2], Wil M.P. van der Aalst[1,2], and Thomas Rose[2,1]

[1] RWTH Aachen University, Germany
{niklas.adams, s.j.v.zelst, wvdaalst}@pads.rwth-aachen.de
[2] Fraunhofer Institute for Applied Information Technology (FIT), Germany
{lara.quack, kathrin.hausmann, thomas.rose}@fit.fraunhofer.de

Abstract. Rapidly changing business environments expose companies to high levels of uncertainty. This uncertainty manifests itself in significant changes that tend to occur over the lifetime of a process and possibly affect its performance. It is important to understand the root causes of such changes since this allows us to react to change or anticipate future changes. Research in process mining has so far only focused on detecting, locating and characterizing significant changes in a process and not on finding root causes of such changes. In this paper, we aim to close this gap. We propose a framework that adds an explainability level onto concept drift detection in process mining and provides insights into the cause-effect relationships behind significant changes. We define different perspectives of a process, detect concept drifts in these perspectives and plug the perspectives into a causality check that determines whether these concept drifts can be causal to each other. We showcase the effectiveness of our framework by evaluating it on both synthetic and real event data. Our experiments show that our approach unravels cause-effect relationships and provides novel insights into executed processes.

Keywords: Process Mining · Concept Drift · Cause-Effect Analysis

1 Introduction

Digitization poses great threats but also exceptional opportunities to companies. On the one hand, new technologies, business models, and legislation expose companies to high levels of uncertainty. On the other hand, the introduction of information systems over the last decades enables companies to collect and analyze data on their *business processes*. These data can be converted into an *event log* and are used to discover, monitor and improve the underlying business processes. It, thus, helps the companies to deal with the uncertainty they are exposed to. *Process mining* [1] is the discipline of computer science that successfully analyzes and improves business processes by applying concepts of process and data science to transform event logs into process models and actionable insight for the process owner. When looking at business processes uncertainty
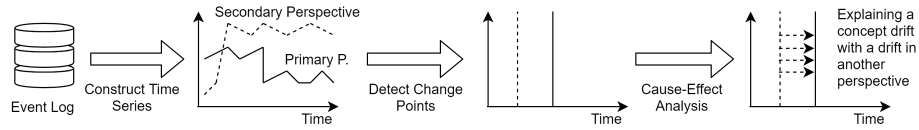
Fig. 1. Our proposed framework transforms event data into two time series representations that can describe various different process perspectives, e.g., the weekly workload. A cause-effect analysis is then conducted using the information of detected concept drifts in the selected perspectives to unravel root causes for these drifts.

is often caused by significant change, called *concept drift*, in some perspective of a business process. For example, due to Covid-19 a lot of companies had to redesign or extend processes by including digital alternatives to previously in-person activities. This resulted in so-called concept drifts. As the quality and profitability of organizations highly depend on their business processes concept drift can have a huge impact on either of these dimensions. The restructuring of a process to meet the health safety regulatory standards could, e.g., lead to increased processing time and thus increased cost. Detecting and handling concept drift has, thus, been named one of the main challenges in process mining [2]. For the process owner the mere knowledge of past occurrences of concept drifts is not sufficient. To derive useful insights it is helpful to know the underlying cause-effect relationships associated with these concept drifts, i.e., the user can either anticipate future concept drifts or use the uncovered relations to further improve the process.

In this paper, we introduce a generic framework that augments concept drift detection in process mining by adding a cause-effect analysis on top of the detected concept drifts. This cause-effect analysis extracts possible explanations for the occurrence of a concept drift. The core idea of the framework is depicted in Figure 1. Before starting, we choose a perspective of the process to be analyzed for concept drift, i.e., the *control-flow*, *data*, *resource* or *performance* perspective. This perspective is called the *primary* perspective. As we are interested in the root causes for these concept drifts, we choose a *secondary* perspective that could contain root causes. This secondary perspective is also analyzed for concept drifts and these are tested for causality with the concept drifts of the primary perspective. In the first step, we transform an event log into two time series for both the chosen primary and secondary perspective. After detecting concept drifts in both perspectives, we conduct a cause-effect analysis and check which concept drifts of the secondary perspective could be causal to a concept drift in the primary perspective. The set of explainable concept drifts forms the output of our framework.

Our framework touches the areas of concept drift detection and cause-effect analysis in process mining which have, thus far, primarily been studied separately. Most of the work on concept drift deals with detecting concept drifts and only considers the control-flow perspective of a process. The control-flow perspective describes the structuring and dependencies of activities. Recent work,

e.g., the approach of Brockhoff et al. [11] introduces additional perspectives, i.e., the time perspective to concept drift detection. Ostovar et al. [27] add an additional characterization of the drift, i.e., providing information about underlying nature of the drift, on top of the mere detection of the drift. With our work, we include more perspectives and add an explainability level to concept drift detection. Most work on cause-effect analysis uncovers cause-effect relationships on a process-instance-level, e.g., giving recommendations for individual customers to maximize the outcome, as Bozorgi et al. [10] recently introduced. Pourbafrani et al. [31] focus on finding cause-effect relationships on a global-process level and use these to simulate what-if scenarios.

The remainder of this paper is structured as follows. We introduce the related work on concept drift detection and cause-effect analysis in Section 2. In Section 3, we provide the definitions and background used in the remainder of this paper. We illustrate our framework for explainable concept drift detection in Section 4. In Section 5, we provide details on our specific implementation and evaluate our framework with synthetic data and conduct a case study on real-life event data. Section 6 concludes the paper.

2   Related Work

A general introduction to the field of process mining is given in [1]. In this section, we introduce related work on concept drift detection and cause-effect analysis in process mining.

Concept drift detection (also: change point detection), has received much attention outside of process mining. A general introduction can be found in [5]. As our use case does not provide labeled data sets for supervised algorithms, we are only interested in unsupervised concept drift detection algorithms as the training data for supervised algorithms is expensive to obtain and the ground truth is hard to define in the setting of real-life event logs. Existing work on concept drift detection in process mining focuses on the detection, localization and characterization of changes, not the explanation of them. Detection refers to the presence of a concept drift, localization to the time of occurrence of the drift and characterization to the nature of the drift, e.g., whether an activity was removed or a performance indicator significantly increased. Explanation refers to the root causes of a drift, e.g., why an activity was removed or why a performance indicator significantly increased. Most of the work aims to detect drifts in the control-flow perspective. Bose et al. [8,9] and Martjushev et al. [24] built representations of the control-flow perspective by using the (directly) follows relations. They employ hypothesis testing to compare a window of values before and after a potential change point for significant differences. Maaradji et al. [22] and Ostovar et al. [27] use the α and α^+-relations [25] to model the control-flow perspective while also using hypothesis testing to determine change points. One notable recent approach is the one of Yeshchenko et al. [38]. The authors use DECLARE constraints to model the control-flow of a process. They define a range of time windows and subsequently calculate values for the declar-

Table 1. Overview of the related approaches for cause-effect analysis and concept drift detection in process mining. Cause-effect analysis can be performed either on the case- or the process level. Approaches for concept drift detection have different scopes, i.e., they can detect, locate, characterize or explain a concept drift. Our approach detects, locates and explains concept drift and therefore yields insights into process level cause-effects.

	Concept Drift				Cause-Effect Analysis	
	Detect.	Locat.	Char.	Expl.	case level	process level
$[4, 8, 9, 11, 12, 21, 22, 24, 36, 39]$	✓	✓				
$[17, 27, 33, 38]$	✓	✓	✓			
$[10, 16, 18, 19, 28, 34]$					✓	
$[26, 29–31]$						✓
Our approach	✓	✓		✓		✓

ative constraints for each time window, forming a multivariate time series. This time series is analyzed for concept drifts by applying the Pruned Exact Linear Time algorithm. By visualizing the results of this clustering, the user can characterize the occurring concept drifts. Other authors also include other perspectives than control-flow. Leontjeva et al. [20] and Meisenbacher et al. [23] use the data payload of past events to include the data perspective into their representation. Analyzing the related work on concept drift in process mining reveals two shortcomings: The little consideration of additional perspectives other than control-flow and the absence of root cause analysis for concept drifts. With this paper we aim to close this gap.

The area of cause-effect analysis in process mining investigates relationships that are present in a process. One way to define different levels of analysis is to either consider the local intra-trace, i.e., case level, or the global level of the process. The case level deals with individual process instantiations, e.g., a customer running through the process of applying for a loan. The global level is the entirety of components and cases that are associated with the process. Many approaches in cause-effect analysis focus on the case level rather than the global level, providing recommendations and predictions for handling individual cases. De Leoni et al. [19] and Hompes et al. [16] provide methods to extract root causes for performance variations on a case level. In another work, Hompes et al. [18] group events based on certain process performance characteristics and further decompose these groups based on different characteristics. They subsequently test for cause-effects between these characteristics by looking at their development over time and testing for Granger-causality [15] to extract the root causes of performance variations on a case level. This technique works well for identifying causal factors for performance variations, however, other perspectives of the process such as control-flow or resources are, so far, not considered, potentially missing important cause-effects. Bozorgi et al. [10] use techniques from causal machine learning to provide recommendations for handling an individual case that maximize the probability of a certain outcome. These approaches provide useful information for individual cases, however, they are not able to detect important cause-effect relationships, that happen on a global level, e.g., an in-

Table 2. Exemplary event log with two cases and resources handling the activities

case-id	activity	timestamp	resource
1	register	2021-06-15 12:30	Peter
1	submit	2021-06-15 12:35	Sophia
2	register	2021-06-15 13:12	Peter
1	reply	2021-06-15 14:21	Christina

crease in customers, that leads to longer waiting times. Other authors investigate cause-effect relationships on a global level. Pourbafrani et al. [30,31] use system dynamics as a modeling tool of the process over time and construct a model that contains cause-effect relationships between different metrics. This model is subsequently used to simulate the outcomes for different scenarios. Nakatumba et al. [26] investigate the effect of resource workload on their performance using regression analysis.

A selection of papers on concept drift and cause-effect analysis in process mining is depicted in Table 1. Our framework is the only technique that covers both spectra.

3   Preliminaries

In this section, we introduce the core definitions of this paper and the main notations used for improving the readability. $\mathcal{P}(X)=\{X'|X'\subseteq X\}$ denotes the power set of a set X. A sequence allows enumerating the elements of a set. A sequence of length n over X is a function $\sigma:\{1,\ldots,n\}\to X$ which we write as $\sigma=\langle\sigma(1),\sigma(2),\ldots,\sigma(n)\rangle$.

An event can be considered the "atomic datum" of process mining. An event consists of values that are assigned to attributes, e.g., the executed activity, the timestamp, a case-id and other attributes. Each line in Table 2 corresponds to an event. Each event needs to be assigned a case-id describing the process instance which is the *case* this event belongs to. All lines with the same case-id in Table 2 form a case. The collection of recorded cases forms an event log.

Definition 1 (Events, Cases and Projections). *An event describes the information associated to the execution of an activity. Let $\mathcal{E}$ denote the universe of events. Let $\mathcal{D}$ denote the universe of attributes and let $\mathcal{V}$ denote the universe of possible attribute values. Let $\mathcal{T}$ denote the universe of possible timestamps.*

- *For an attribute $d \in \mathcal{D}$ we assume the existence of a mapping to retrieve the corresponding attribute value $\pi_d^{\mathcal{E}}:\mathcal{E}\nrightarrow\mathcal{V}$.*
- *The activity projection is a total function retrieving the activity of an event $\pi_{act}^{\mathcal{E}} : \mathcal{E} \to \mathcal{A}$, where $\mathcal{A}$ denotes the universe of activities.*
- *The time projection is a total function retrieving the timestamp of an event $\pi_{time}^{\mathcal{E}}:\mathcal{E}\to\mathcal{T}$.*
- *Each event has an identifier $\pi_{id}^{\mathcal{E}}$ to differentiate between events, that might have the same values for each attribute. Therefore, $e,e'\in\mathcal{E}(\pi_{id}^{\mathcal{E}}(e)=\pi_{id}^{\mathcal{E}}(e') \Rightarrow e=e')$*
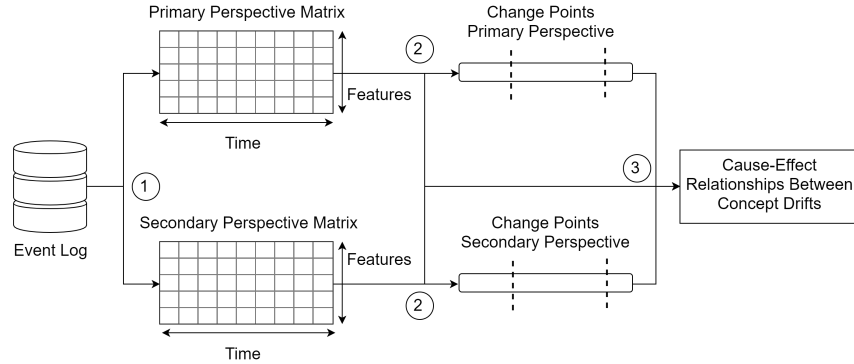
Fig. 2. Our proposed framework for uncovering cause-effect relationships. In Step 1, the event log is transformed into two time series representations of different perspectives. In Step 2, the change points are detected and checked for causality in Step 3.

Events belong to a case denoting the process instance of this event. Let $\mathcal{C}$ be the universe of all cases.

- *For an attribute $d \in \mathcal{D}$ we assume the existence of a projection function to retrieve the corresponding attribute value $\pi_d^{\mathcal{C}} : \mathcal{C} \nrightarrow \mathcal{V}$.*
- *Each event $e \in \mathcal{E}$ has a case-id describing the case it belongs to. The projection function $\pi_{case}^{\mathcal{E}}(e)$ retrieves the corresponding case-id.*
- *We furthermore assume the existence of an event projection that maps a case on the set of its events $\pi_{events}^{\mathcal{C}} : \mathcal{C} \rightarrow \mathcal{P}(\mathcal{E})$. Cases are non-overlapping, i.e., $\forall c_1, c_2 \in \mathcal{C}(\pi_{events}^{\mathcal{C}}(c_1) \cap \pi_{events}^{\mathcal{C}}(c_2) \neq \emptyset \Rightarrow c_1 = c_2)$.*

An event log L is a set of cases, thus $L \in \mathcal{P}(\mathcal{C})$.

- *For an attribute $d \in \mathcal{D}$ we assume the existence of the projection functions $\pi_d^{\mathcal{E}}(L) = \{\pi_d^{\mathcal{E}}(e) | \exists_{c \in L} e \in \pi_{events}(c)\}$ and $\pi_d^{\mathcal{C}}(L) = \{\pi_d^{\mathcal{C}}(c) | c \in L\}$ to retrieve the set of values for this attribute.*

4   Generic Framework for Explainable Concept Drift

In this section, we introduce the three steps of our framework. The framework is depicted in Figure 2. In the first step, we construct time series representations for both chosen perspectives. In the second step, a change point detection algorithm is performed on both perspectives. In the third step, we test pairs of change points for causality by taking the time lag between them and check whether the two time series can be causal given the lag. If this test is positive the detected concept drift together with the cause-effect-relationship are as explanation given to the user. For each of the steps, we provide the input and output specifications.

4.1 Time Series Construction

To express the development of a process perspective over time, we construct a time series. A time series assigns values to subsequent time intervals. We, therefore. need to map an event log onto time intervals and then assign values to these intervals. For assigning values to a set of events that are contained in a specific time interval, we first need a way to look at these events in isolation and thus define a selection function for events based on a time interval.

Definition 2 (Time Intervals). *Based on a reference timestamp $t_r \in \mathcal{T}$, e.g., describing the beginning of an event log, we can express every timestamp as a real number that describes the time, e.g., number of milliseconds or nanoseconds, passed since this reference timestamp. The real-valued representation of a timestamp can be retrieved with the function $r_{t_r}(t) \in \mathbb{R}$. Given two timestamps $t_1, t_2 \in \mathcal{T}$ and $t_1^{\mathbb{R}} = r_{t_r}(t_1), t_2^{\mathbb{R}} = r_{t_r}(t_2)$ we define a time interval $ti = [t_1^{\mathbb{R}}, t_2^{\mathbb{R}})$ where $t_1^{\mathbb{R}} < t_2^{\mathbb{R}}$. Let $\mathcal{TI}$ denote the universe of possible time intervals. $|ti| = t_2^{\mathbb{R}} - t_1^{\mathbb{R}}$ defines the length of a time interval and $start(ti) = t_1^{\mathbb{R}}$. To extract the events of an event log L occurring within a time interval $ti \in \mathcal{TI}$ we define the selection function $sel(L, ti) = \{e \in \mathcal{E} | \exists_{c \in L} e \in \pi_{events}^{\mathcal{C}}(c) \wedge r_{t_r}(\pi_{time}^{\mathcal{E}}(e)) \in ti\}$.*

This definition is used to map an event log onto time intervals. To finish constructing the time series we now calculate a value for each time interval. We first need to define a function to map a collection of events onto a real-valued number and can then extend this to multiple functions, mapping a collection of events onto a real-valued vector and finally a time series, doing this for multiple subsequent time intervals.

Definition 3 (Event Log to Time Series). *Given a time interval $ti \in \mathcal{TI}$ and an event log L, we define the function $f(sel(L, ti)) \in \mathbb{R}$ that maps an event log for a specific time interval onto a real-valued number. We use the notation $f(L, ti) = f(sel(L, ti))$ for readability. Let $f_1, f_2, \ldots, f_m$ be functions of the signature of f. We define the function $g(L, ti) = (f_1(L, ti), f_2(L, ti), \ldots, f_m(L, ti))$ with $g(L, ti) \in \mathbb{R}^m$ to construct a real-valued vector for a specific time interval of an event log. Let $TI \in \mathcal{TI}^*$ define a mutually exclusive sequence of time intervals of equal lengths, i.e., $\forall ti_i, ti_j \in TI(|ti_i| = |ti_j| \wedge i \neq j \Rightarrow ti_i \cap ti_j = \emptyset \wedge i < j \Rightarrow start(ti_i) < start(ti_j))$. With $TI = \langle ti_1, \ldots, ti_n \rangle$ we define the time series construction function $h_{g,TI,L} = (g(L, ti_1)^\intercal, \ldots, g(L, ti_n)^\intercal)$ with $h_{g,TI,L} \in \mathbb{R}^{m \times n}$ to retrieve a real-valued matrix, that represents a multivariate time series of an event log.*

For each perspective there are many ways to represent it as a real-valued number. Take the control-flow as an example. We can count the number of distinct activities for subsequent time intervals. If the number of activities suddenly increases we know that there is a concept drift in the control-flow since a new activity was added. Different measurements can be combined as a single one can not represent the whole perspective. E.g., if one activity was removed and one was added taking the number of distinct activities as representation could not express this. Table 3 depicts a non-exhaustive list of measurements for each of

Table 3. Possible mapping functions to construct a real-valued representation of different perspectives of a business process.

Control-Flow	Performance	Data	Resources
Directly-follows frequencies [3]	Service times [26]	Aggregation of case attributes	Workload [26]
α-relations [3]	Overtime cases [34]	Aggregation of activity attributes	Involved resources
α^+-relations [25]	Case durations [18]	Number of events or cases	Number of active resources
Heuristic Miner's $a \Rightarrow_W b$ score [37]	Activity sojourn time [18]	Threshold exceedings	Aggregation of attribute values
Number of activities	Activity waiting time [18]		
DECLARE constraints [38]			
...	...	...	...

the introduced perspectives. For a detailed introduction we refer to the corresponding papers.

For the control-flow perspective, we can use simple measures as, e.g., the number of distinct activities or intermediate results of mining algorithms like the α-relations obtained from the α-Miner [3].

For the performance perspective we can leverage heavily on the recorded times, which can be seen as a proxy for cost or service quality. We can, e.g., calculate the average service times for each activity, i.e., the time from start to completion of an activity. Furthermore, we can define a threshold of processing time and count all the cases that exceed this threshold and are thus classified as overtime.

The measures for the data perspective use the additional attributes associated to events, e.g., the age or credit score of customers. We can use aggregation functions such as average or maximum to map all the values of an attribute in a time interval onto a single number, e.g., the average age of customers for each time interval. We can, furthermore, count the number of events to describe the event volume over time.

Representations for the resource perspective rely on information about the resources, often staff members, handling an activity. We can count the number of events a resource is involved in to calculate the workload and its development over time. By simply counting the number of active resources each time frame, we can furthermore monitor the number of deployed resources over time.

The question remains which perspectives a user should choose. There is not a general answer for this, domain knowledge and potential assumptions can be used. However, the investigation of certain perspectives might be more promising than others. There are some examples of reoccurring cause-effect themes in process mining. Resources often have an impact onto the performance of a process, e.g., the workload onto the service times [26], the workload onto overtime cases [34] or the associated data onto the case duration [28]. Furthermore, e.g., control-flow changes such as changing prevalence of a choice might influence the performance perspective.

4.2 Change Point Detection

After constructing multivariate time series for each the primary and the secondary perspective, we detect change points in these time series. The change point detection technique maps a time series onto subsets of the time intervals in which the distribution of the features significantly changed.

Definition 4 (Change Point Detection in Multivariate Time Series). *Let $H \in \mathbb{R}^{m \times n}$ be a multivariate time series and $TI = \langle ti_1, \ldots, ti_n \rangle$ be the previously introduced sequence of time intervals used to construct this time series. A change point detection technique CPD maps a time series onto a subset of time intervals, where a significant change of the underlying time series occurred $CPD(H) \subseteq TI$.*

As an input the change point detection method has to be able to process a multivariate time series. As mentioned in Section 2, this method should be unsupervised, i.e., be able to detect change points without seeing similar kinds of time series with annotated change points before. Examples of change point detection techniques include cost-based techniques [35], hypothesis testing [9] or clustering techniques [17].

4.3 Cause-Effect Analysis

In the first step, two time series for the primary and the secondary perspective are constructed. A set of m_p mapping functions for the primary perspective and m_s for the secondary perspective are applied to construct the time series $H_p \in \mathbb{R}^{m_p \times n}$ and $H_s \in \mathbb{R}^{m_s \times n}$. The change point detection step of the framework calculates two sets of change points $CPD(H_p)$ and $CPD(H_s)$. In the cause-effect analysis step, we look at the change points in the primary perspective and analyse which concept drifts in the secondary perspective potentially have a cause-effect relationship to a concept drift in the primary perspective. We, therefore, look at each primary drift and consider all preceding secondary drifts. We calculate the time lag, i.e., the number of time intervals that lie between the drifts, and test whether the secondary perspective can potentially be causal to the primary perspective given this lag. We, therefore, test all pairs of features between primary and secondary perspective for causality given this lag. A feature is a row of the time series describing a single measurement over time. If a feature pair is tested to be causal we add the change point pair and all causal feature pairs to the output of our framework.

Definition 5 (Cause-Effect Analysis). *Let $H_p \in \mathbb{R}^{m_p \times n}$ and $H_s \in \mathbb{R}^{m_s \times n}$ be time series for the primary and secondary perspective. We define the lag function $l_{TI}(ti_1, ti_2) \in \mathbb{N}$ to retrieve the number of time intervals in TI that lie in between ti_1 and ti_2, i.e., the lag. For a change point of the primary perspective $cp_p \in CPD(H_p)$ and a change point in the secondary perspective $cp_s \in CPD(H_s)$ we retrieve the lag k using the lag function $k = l_{TI}(cp_p, cp_s)$. Given a row, i.e., a feature, of the primary and secondary perspective $h_p \in H_{p,i}, i \in \{1, \ldots, m_p\}$*

and $h_s \in H_{s,i}, i \in \{1, \ldots, m_s\}$, where $h_s, h_p \in \mathbb{R}^n$, a cause-effect analysis technique CA maps these two features and a time lag k onto a value between 0 and 1 $CA(h_p, h_s, k) \in [0, 1]$. This value indicates whether a cause-effect relationship with lag k is present or not.

The set of all change point pairs with all detected cause-effect relationships between feature pairs forms the output of the framework.

5 Evaluation

5.1 Implementation

We implemented our framework on the basis of PM4Py [7]. The implemented version is available at GitHub[3]. In this section, we introduce the techniques in change point detection and cause-effect analysis specific to our implementation.

Similar to Yeshchenko et al. [38], we use the Pruned Exact Linear Time PELT-algorithm [14] as a change point detection technique for multivariate time series. This technique uncovers change points by minimizing a cost function that depends on assigning change points. It is able to process a multivariate time series and computes an optimal solution in linear time and is therefore well suited for our experiments. An exact description can be found in [14]. For applying the PELT-algorithm a penalty β has to be chosen that prevents overfitting. The calculated change points are subsequently used to calculate the lags needed for cause-effect analysis. We use the concept of Granger-causality [15]. Granger-causality determines with which probability two time series are correlated given a time lag between them and can thus be seen as a type of predictive causality. The user has to provide a p-value that describes the threshold probability at which feature pairs should be classified as Granger-causal.

5.2 Experiments

We evaluate our framework using a synthetic event log and a real-life event log. The synthetic log is used as a means to verify the implementation and a proof of concept. We, then, expand this to conduct a case study on real-life event data and discuss our findings. For both experiments we provide the chosen parameters for the three steps of our framework, i.e., perspectives and measurements in time series construction, change point detection in multivariate time series and cause-effect analysis. To verify our results, we compare our findings to state-of-the-art methods in concept drift detection and cause-effect analysis.

5.3 Synthetic Insurance Event Log

We use CPN Tools [32] to generate an event log based on a simulation model of claiming insurance shown in Figure 3. The different ways of notification, i.e.

---

[3] `https://github.com/niklasadams/explainable_concept_drift_pm.git`
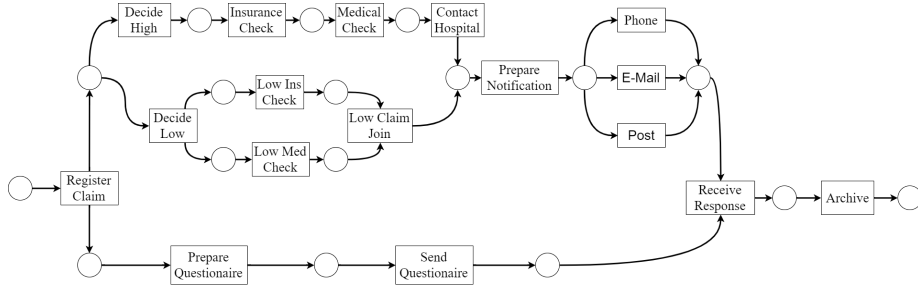
Fig. 3. Synthetic process for claiming insurance. The availability of different ways of notification, phone, e-mail and post, depends on the age of the customer. [8]

Table 4. Parameter choices for running the framework on the synthetic event log.

Experimental Setup			
Time series construction	Primary perspective	Control-Flow	Directly-follows frequencies
	Secondary perspective	Data	Minimum, maximum, sum, average, count, set average
	Time interval duration	1 day	
Change point detection	PELT-algorithm	$\beta_{primary} = 3$	$\beta_{secondary} = 1.5$
Cause-effect analysis	Granger-causality	p-value=1×10^{-12}	

postal, phone and email, are available for different ages of customers. When simulating we introduce a drift in the ages of customers that should then cause a succeeding drift in the prevalence of notification activities, especially an increase in email-notification. The chosen parameters for the instantiation of our framework are depicted in Table 4.

For change point detection we retrieve a change point in the primary control-flow perspective at day 133 and in the secondary data perspective at day 132. The lag between this drift is $k=1$. We, therefore, use a lag of 1 when testing for a cause-effect relationship between the two perspectives.

With a p-value of 10^{-12}, which is especially low due to the artificial setting, we retrieve 25 feature pairs that are Granger-causal with lag $k=1$. All involved features of the primary control-flow perspective concern the frequency of directly follows relationships between one of the notification activities and either a preceding or succeeding activity. The features of the secondary perspective all describe the distribution of age, i.e., the sum, average, minimum, maximum and average of the set of values. We, therefore, limit our output to only 5 of the feature pairs, which are depicted in Figure 4. The depicted features propose that a decrease in the age of customers led to an increase in the prevalence of the email notification activity one day later which is exactly the cause-effect relationship we artificially introduced. Our framework has correctly detected and explained the concept drift with the underlying cause-effect relationship.
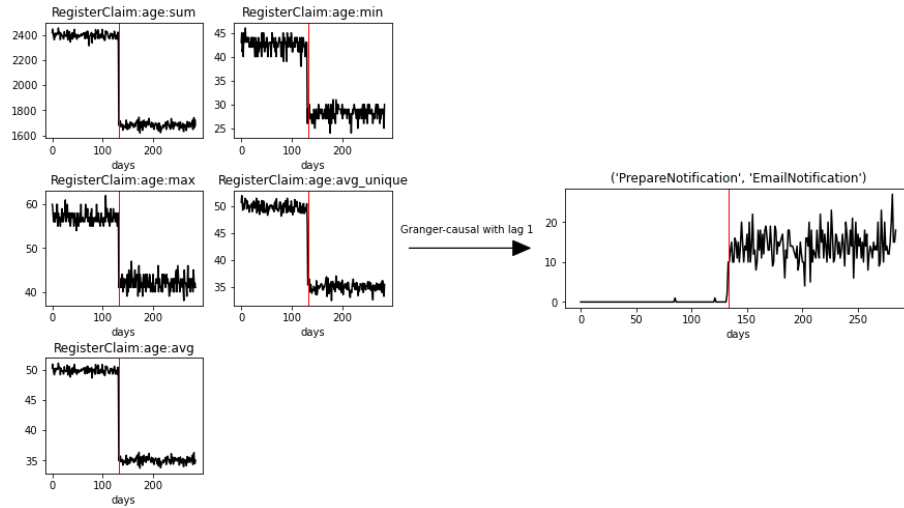
Fig. 4. 5 Granger-causal feature pairs for the cause-effect relationship between data and control-flow perspective. A drift in the age of the customers is responsible for an increase in email notifications and a decrease in other notifications.

5.4   Case Study

We also evaluated our framework using data from the Business Process Intelligence (BPI) Challenge 2017 [13]. The event log considered belongs to a loan application process through an online system. A customer can submit loan applications to the financial institute and may receive an offer from the financial institute afterwards. The parameters for applying our framework are depicted in Table 5. We analyze the performance perspective, i.e., the service times, for concept drifts. We search for root causes in the resource perspective, i.e., the workload, as it has shown to often have a significant impact on the service times [26].

For the primary, performance perspective we retrieve a change point in Week 28. For the secondary, workload perspective we retrieve a change point at Week 22. The lag for cause-effect analysis is therefore k=6. The cause-effect analysis with a p-value of 0.015 yields 23 Granger-causal feature pairs with a lag of 6 weeks. Four different primary features are contained in these feature pairs. Since three of them do not exhibit a concept drift around week 28, we drop the corresponding pairs for further analysis. The remaining pairs are depicted in Figure 5. The average duration of *W_Validate application* shows a significant decrease for week 28. We further analyze the resource workloads that are Granger-causal to this feature. Most of the resources do not work continuously over the span of the event log. We can see increases and peaks in the workload for some resource. When looking at the total workload of all resources, which is among the Granger-causal features, we can see a significant increase. The detected cause-effect relationship, therefore, states that **an increase in**

Table 5. Parameter choices for running the framework on the BPI 2017 log.

Experimental Setup			
Time series construction	Primary perspective	Performance	Service times [26]
	Secondary perspective	Resource	Workload [26]
	Time interval duration	1 week	
Change point detection	PELT-algorithm	$\beta_{primary} = 3$	$\beta_{secondary} = 1.5$
Cause-effect analysis	Granger-causality	p-value=0.015	
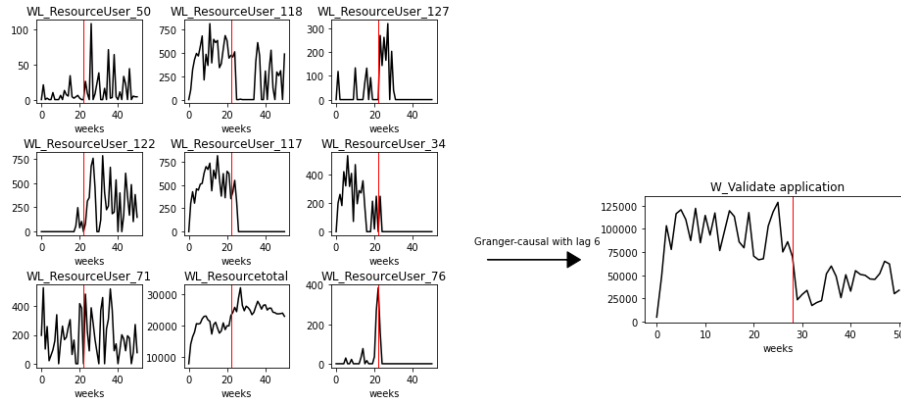
Fig. 5. Granger-causal feature pairs of the cause-effect relationship between resource and performance perspective. The only concept drift for the duration perspective of these features can be observed in *W_Validate application*. An increase of the workload is Granger-causal to the reduction of the service times.

the workload of resources led to a decrease in the service times for *W_Validate application*.

One submission paper for the BPI Challenge [6], amongst other things, investigates different KPIs of the process over time. This paper also found a significant decrease in the manual validation time, i.e., the service times for validating an application, and an increase in case numbers. Due to the absence or the lack of visibility of other factors such as additional training, change of staff, etc., the paper suggests that the decrease in service times is a reaction to cope with the increased workload.

5.5   Comparison

Our proposed framework for explainable concept drift detection touches two areas of research: Concept drift detection and cause-effect analysis. We, therefore, compare the results for the synthetic event log and the BPI 2017 log with the results from state-of-the-art methods from both of these areas. For concept

Table 6. Comparison of results for the concept drift detection

	Our Approach	Visual Analytics [38]	ProDrift [22]
Synthetic Log	Control-flow drift, day 133	✓	✓
BPI 2017	Performance drift, week 28	✗	✗

Table 7. Comparison of results for cause-effect analysis

	Our Approach	PMSD [29]
Synthetic Log	Data (age) → control-flow	✗
BPI 2017	Resource (workload) → performance	✓

drift detection we compare the results with the visual analytics approach of Yeshchenko et al. [38] and ProDrift by Maaradji et al. [21] as both approaches have shown outstanding results in concept drift detection. For cause-effect analysis we compare our results with the findings of Pourbafrani et al. [29] as they are searching for relations between different process parameters on a system-wide level.

Table 6 depicts the comparison between the detected concept drifts for the synthetic and the real-life event log. The control-flow drift in the synthetic log is detected by both approaches. As ProDrift relies on completed traces, the drift is detected approximately 15 days later compared to our approach. Both approaches very clearly show the existence of a sudden drift through means of their visualization. As both approaches do only focus on control-flow drifts they can not be used to compare results on the detected performance drift for the BPI 2017 log. Table 7 depicts the results retrieved from PMSD compared to our approach. As PMSD does not model the data perspective, we can not use it to detect the cause-effect in the synthetic log. For the BPI 2017 log we apply the PMSD framework and compute a system dynamics log. This log contains, among others, the arrival rate and the service times of the process. We apply the relation detection of PMSD with a lag of 6 weeks. The results show a negative correlation between the lagged arrival rate and the service times. This corresponds to our detected cause-effect as the higher influx of cases determined a decrease in service times.

We verified our findings by applying state-of-the-art methods from both concept drift detection and cause-effect analysis. If the corresponding perspective can be modeled, we are able to verify our findings with these approaches. These are promising results as they show the power of incorporating more perspectives into concept drift detection and using these to find possible cause-effects of concept drifts.

6   Conclusion

In this paper, we combine concept drift detection and cause-effect analysis to create a framework for explainable concept drift detection. We define a primary perspective where concept drifts should be detected and a secondary perspective

with which these concept drifts are explained. By applying a cause-effect analysis to the features of both perspectives, we identify feature pairs that can be used to explain the concept drift. We verified our approach using a synthetically generated event log. We furthermore analyzed the event log of the BPI Challenge 2017 and were able to explain a concept drift in the performance with an increase in the resources' workload. These first experiments have shown a great potential for explaining concept drifts.

Future Work To further improve our conceptual framework the following steps can be taken. First of all, we want to plug different change point detection algorithms and cause-effect analysis tools to detect other types of drifts and, e.g., non-linear relationships. Furthermore, spurious elements and rare signals produce spikes in a signal that can be misleading to cause-effect analysis techniques. We want to investigate whether the general application of noise filtering on the time series is beneficial. Another interesting point for an extension of the framework are non pairwise dependencies. A concept drift could, e.g., be caused by two different concept drift in two other perspective and not by only one of them.

References

1. van der Aalst, W.M.P.: Process mining: Data science in action. Springer (2016)
2. van der Aalst, W.M.P., et al.: Process mining manifesto. In: BPM. vol. 99, pp. 169–194. Springer (2011)
3. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
4. Accorsi, R., Stocker, T.: Discovering workflow changes with time-based trace clustering. In: SIMPDA. pp. 154–168. Springer (2011)
5. Aminikhanghahi, S., Cook, D.: A survey of methods for time series change point detection. Knowl. Inf. Syst. **51**, 339–367 (2017)
6. Berger, F.: Mining event log data to improve a loan application process. BPI Challenge (2017), `https://www.win.tue.nl/bpi/lib/exe/fetch.php?media=2017:bpi2017_paper_3.pdf`
7. Berti, A., van Zelst, S.J., van der Aalst, W.M.P.: Process mining for python (PM4PY): Bridging the gap between process-and data science. arXiv preprint arXiv:1905.06169 (2019)
8. Bose, R.P.J.C., van der Aalst, W.M.P., Žliobaitė, I., Pechenizkiy, M.: Handling concept drift in process mining. CAiSE pp. 391–405 (2011)
9. Bose, R.P.J.C., van der Aalst, W.M.P., Žliobaitė, I., Pechenizkiy, M.: Dealing with concept drifts in process mining. IEEE Trans. Neural Networks Learn. Syst. **25**(1), 154–171 (2014)
10. Bozorgi, Z.D., Teinemaa, I., Dumas, M., Rosa, M.L., Polyvyanyy, A.: Process mining meets causal machine learning: Discovering causal rules from event logs. In: ICPM. pp. 129–136 (2020)
11. Brockhoff, T., Uysal, M.S., van der Aalst, W.M.P.: Time-aware concept drift detection using the earth mover's distance. In: ICPM. pp. 33–40 (2020)

12. Carmona, J., Gavaldà, R.: Online techniques for dealing with concept drift in process mining. In: IDA. vol. 7619, pp. 90–102 (2012)
13. van Dongen, B.: BPI Challenge 2017
14. Gachomo, D.: The power of the pruned exact linear time (PELT) test in multiple changepoint detection. American Journal of Theoretical and Applied Statistics **4**, 581 (2015)
15. Granger, C.: Some recent development in a concept of causality. Journal of Econometrics **39**(1), 199 – 211 (1988)
16. Hompes, B., Buijs, J.C.A.M., van der Aalst, W.M.P.: A generic framework for context-aware process performance analysis. In: OTM. vol. 10033, pp. 300–317 (2016)
17. Hompes, B., Buijs, J.C.A.M., van der Aalst, W.M.P., Dixit, P.M., Buurman, H.: Detecting change in processes using comparative trace clustering. In: SIMPDA. vol. 1527, pp. 95–108 (2015)
18. Hompes, B.F.A., Maaradji, A., Rosa, M.L., Dumas, M., Buijs, J.C.A.M., van der Aalst, W.M.P.: Discovering causal factors explaining business process performance variation. In: CAiSE 2017. vol. 10253, pp. 177–192 (2017)
19. de Leoni, M., van der Aalst, W.M.P., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Inf. Syst. **56**, 235–257 (2016)
20. Leontjeva, A., Conforti, R., Francescomarino, C.D., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: BPM. vol. 9253, pp. 297–313 (2015)
21. Maaradji, A., Dumas, M., Rosa, M.L., Ostovar, A.: Fast and accurate business process drift detection. In: BPM. vol. 9253, pp. 406–422 (2015)
22. Maaradji, A., Dumas, M., Rosa, M.L., Ostovar, A.: Detecting sudden and gradual drifts in business processes from execution traces. IEEE Trans. Knowl. Data Eng. **29**(10), 2140–2154 (2017)
23. Maisenbacher, M., Weidlich, M.: Handling concept drift in predictive process monitoring. In: SCC. pp. 1–8 (2017)
24. Martjushev, J., Bose, R.P.J.C., van der Aalst, W.M.P.: Change point detection and dealing with gradual and multi-order dynamics in process mining. In: BIR. vol. 229, pp. 161–178 (2015)
25. de Medeiros, A.K.A., van Dongen, B.F., van der Aalst, W.M.P., Weijters, A.J.M.M.: Process mining : Extending the α-algorithm to mine short loops. In: BETA Working Paper Series. vol. WP 113 (2004)
26. Nakatumba, J., van der Aalst, W.M.P.: Analyzing resource behavior using process mining. In: BPM. vol. 43, pp. 69–80 (2009)
27. Ostovar, A., Maaradji, A., Rosa, M.L., ter Hofstede, A.H.M.: Characterizing drift from event streams of business processes. In: CAiSE. vol. 10253, pp. 210–228 (2017)
28. Polato, M., Sperduti, A., Burattin, A., de Leoni, M.: Data-aware remaining time prediction of business process instances. In: IJCNN. pp. 816–823 (2014)
29. Pourbafrani, M., van der Aalst, W.M.P.: PMSD: Data-driven simulation using system dynamics and process mining. In: BPM (PhD/Demos). vol. 2673, pp. 77–81 (2020)
30. Pourbafrani, M., van Zelst, S.J., van der Aalst, W.M.P.: Scenario-based prediction of business processes using system dynamics. In: OTM. vol. 11877, pp. 422–439 (2019)
31. Pourbafrani, M., van Zelst, S.J., van der Aalst, W.M.P.: Supporting automatic system dynamics model generation for simulation in the context of process mining. In: BIS. vol. 389, pp. 249–263 (2020)

32. Ratzer, A.V., et al.: CPN tools for editing, simulating, and analysing coloured petri nets. In: ICATPN. vol. 2679, pp. 450–462 (2003)
33. Seeliger, A., Nolle, T., Mühlhäuser, M.: Detecting concept drift in processes using graph metrics on process graphs. In: S-BPM ONE. p. 6 (2017)
34. Suriadi, S., Ouyang, C., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Root cause analysis with enriched process logs. In: BPMs. vol. 132, pp. 174–186 (2012)
35. Truong, C., Oudre, L., Vayatis, N.: Selective review of offline change point detection methods. Signal Processing **167**, 107299 (2020)
36. Weber, P., Bordbar, B., Tiño, P.: Real-time detection of process change using process mining. In: ICCSW. vol. DTR11-9, pp. 108–114 (2011)
37. Weijters, A., van der Aalst, W.M.P., Medeiros, A.: Process mining with the heuristics miner-algorithm. CIRP Annals-Manufacturing Technology **166** (2006)
38. Yeshchenko, A., Ciccio, C.D., Mendling, J., Polyvyanyy, A.: Comprehensive process drift detection with visual analytics. In: ER. vol. 11788, pp. 119–135 (2019)
39. Zheng, C., Wen, L., Wang, J.: Detecting process concept drifts from event logs. In: OTM. vol. 10573, pp. 524–542 (2017)