Dealing With Concept Drifts in Process Mining

R. P. Jagadeesh Chandra Bose, Wil M. P. van der Aalst, Indrė Žliobaitė, and Mykola Pechenizkiy

***Abstract*—Although most business processes change over time, contemporary process mining techniques tend to analyze these processes as if they are in a steady state. Processes may change suddenly or gradually. The drift may be periodic (e.g., because of seasonal influences) or one-of-a-kind (e.g., the effects of new legislation). For the process management, it is crucial to discover and understand such concept drifts in processes. This paper presents a generic framework and specific techniques to detect when a process changes and to localize the parts of the process that have changed. Different features are proposed to characterize relationships among activities. These features are used to discover differences between successive populations. The approach has been implemented as a plug-in of the ProM process mining framework and has been evaluated using both simulated event data exhibiting controlled concept drifts and real-life event data from a Dutch municipality.**

***Index Terms*—Concept drift, flexibility, hypothesis tests, process changes, process mining.**

I. Introduction

BUSINESS processes are nothing more than logically related tasks that use the resources of an organization to achieve a defined business outcome. Business processes can be viewed from a number of perspectives, including the control flow, data, and the resource perspectives. In today's dynamic marketplace, it is increasingly necessary for enterprises to streamline their processes so as to reduce cost and to improve performance. In addition, today's customers expect organizations to be flexible and adapt to changing circumstances. New legislations such as the WABO act [1] and the Sarbanes–Oxley Act [2], extreme variations in supply and demand, seasonal effects, natural calamities and disasters, deadline escalations [3], and so on, are also forcing organizations to change their processes. For example, governmental and insurance organizations reduce the fraction of cases being checked when there is too much of work in the pipeline. As another example, in a disaster, hospitals, and banks change their operating procedures. It is evident that the economic success of an organization is more and more dependent on its ability to react and adapt to changes in its operating environment. Therefore, flexibility and change have been studied in-depth in the context of business process management (BPM). For example, process-aware information systems (PAISs) [4] have been extended to be able to flexibly adapt to changes in the process. State-of-the-art workflow management (WFM) and BPM systems [5] provide such flexibility, e.g., we can easily release a new version of a process. In addition, in processes not driven by WFM/BPM systems (such as the usage of medical systems) there is even more flexibility as processes are controlled by people rather than information systems.

Many of today's information systems are recording an abundance of event logs. Process mining is a relatively young research discipline aimed at discovering, monitoring, and improving real processes by extracting knowledge from event logs [6] (Section II-A for a brief introduction). Although flexibility and change have been studied in-depth in the context of WFM and BPM systems, contemporary process mining techniques assume the processes to be in a steady state. For example, when discovering a process model from event logs, it is assumed that the process at the beginning of the recorded period is the same as the process at the end of the recorded period. Using ProM,[1] we have analyzed processes in more than 100 organizations. These practical experiences show that it is very unrealistic to assume that the process being studied is in a steady state. As mentioned earlier, processes may change to adapt to changing circumstances. Concept drift refers to the situation in which the process is changing while being analyzed. There is a need for techniques that deal with such second-order dynamics. Analyzing such changes is of utmost importance when supporting or improving operational processes and to obtain an accurate insight on process executions at any instant of time. When dealing with concept drifts in process mining, the following three main challenges emerge.

1) *Change point detection:* The first and most fundamental problem is to detect concept drift in processes, i.e., to detect that a process change has taken place. If so, the next step is to identify the time periods at which changes have taken place. For example, by analyzing an event log from an organization (deploying seasonal processes), we should be able to detect that process changes happen and that the changes happen at the onset of a season.
2) *Change localization and characterization:* Once a point of change has been identified, the next step is to characterize the nature of change, and identify the region(s) of change (localization) in a process. Uncovering the nature of change is a challenging problem that involves both the identification of change perspective (e.g., control flow, data, resource, sudden, gradual, and so on) and the identification of the exact change itself. For instance, in the example of a seasonal process, the change could be

Manuscript received May 15, 2012; revised June 24, 2013; accepted August 2, 2013.

R. P. J. C. Bose, W. M. P. van der Aalst, and M. Pechenizkiy are with the Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven 5600 MB, The Netherlands (e-mail: j.c.b.rantham.prabhakara@tue.nl; w.m.p.v.d.aalst@tue.nl; m.pechenizkiy@tue.nl).

I. Žliobaitė is with the Department of Information and Computer Science, Aalto University, Aalto FI-00076, Finland (e-mail: indre.zliobaite@aalto.fi).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2013.2278313

[1] See www.processmining.org for more information.

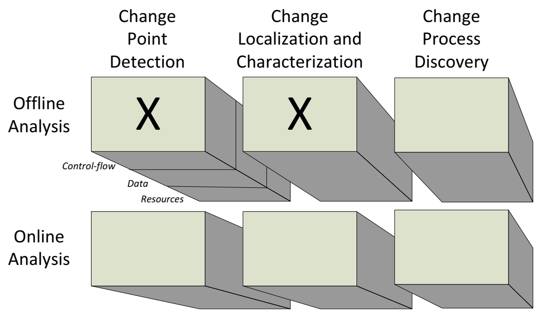


Fig. 1. Different dimensions of concept drift analysis in process mining.

that more resources are deployed or that special offers are provided during holiday seasons.

3) *Change process discovery:* Having identified, localized, and characterized the changes, it is necessary to put all of these in perspective. There is a need for techniques/tools that exploit and relate these discoveries. Unraveling the evolution of a process should result in the discovery of the change process describing the second-order dynamics. For instance, in the example of a seasonal process, we could identify that the process recurs every season. In addition, we can show an animation on how the process evolved over a period with annotations showing several perspectives such as the performance metrics (service levels, throughput time, and so on) of a process at different instances of time.

We can differentiate between two broad classes of dealing with concept drifts when analyzing event logs (Fig. 1).

1) *Offline analysis:* This refers to the scenario where the presence of changes or the occurrence of drifts need not be uncovered in a real time. This is appropriate in cases where the detection of changes is mostly used in postmortem analysis, the results of which can be considered when designing/improving processes for later deployment. For example, offline concept drift analysis can be used to better deal with seasonal effects (hiring less staff in summer or skipping checks in the weeks before Christmas).
2) *Online analysis:* This refers to the scenario where changes need to be discovered in near real time. This is appropriate in cases where an organization would be more interested in knowing a change in the behavior of their customers or a change in demand as and when it is happening. Such real-time triggers (alarms) will enable organizations to take quick remedial actions and avoid any repercussions.

In this paper, we focus on two of the challenges: 1) change (point) detection and change localization and 2) characterization in an offline setting (Fig. 1). We define different features and propose a framework for dealing with these two problems from a control-flow perspective. Initially, we show the promise of the techniques proposed in this paper on a synthetic log and later evaluate them on a real-life case study from a large Dutch municipality.

The rest of this paper is organized as follows. Section II provides background on process mining and concept drifts in data mining. Related work is presented in Section III. Section IV describes the various aspects and nature of change, whereas Section V presents the basic idea for change detection in event logs. Section VI introduces various features that capture the characteristics of event logs. Section VII illustrates the significance of statistical hypothesis tests for detecting drifts. Section VIII presents the framework for dealing with concept drifts in process mining, whereas Section IX presents the realization of the proposed approaches in the ProM framework. Section X describes the effectiveness of the features and the techniques proposed in this paper on a synthetic log as well as a real-life case study. Finally, this paper is summarized with a conclusion and an outlook on some of the open research questions in Section XI.

II. Background

In this section, we discuss the basic concepts in process mining and concept drifts in data mining/machine learning.

A. Process Mining

Process mining serves a bridge between data mining and business process modeling [6]. Business processes leave trails in a variety of data sources (e.g., audit trails, databases, and transaction logs). Process mining aims at discovering, monitoring, and improving real processes by extracting knowledge from event logs recorded by a variety of systems (ranging from sensor networks to enterprise information systems). The starting point for process mining is an event log, which is a collection of events. We assume that events can be related to process instances (often called cases) and are described by some activity name. The events within a process instance are ordered. Therefore, a process instance is often represented as a trace over a set of activities. In addition, events can have attributes such as timestamps, associated resources (e.g., the person executing the activity), transactional information (e.g., start, complete, suspend, and so on), and data attributes (e.g., amount or type of customer). For a more formal definition of event logs used in process mining, the reader is referred to [6]. Fig. 2 shows a fragment of an example log. Event logs like in Fig. 2 are completely standard in the process mining community and event log formats such as MXML [7] and XES [8] are used.

The topics in process mining can be broadly classified into three categories: 1) discovery; 2) conformance; and 3) enhancement [6]. Process discovery deals with the discovery of models from event logs. These models may describe control flow, organizational aspects, time aspects, and so on. For example, there are dozens of techniques that automatically construct process models (e.g., Petri nets or BPMN models) from event logs [6]. Fig. 2 shows the basic idea of process discovery. An event log containing detailed information about events is transformed into a multiset of traces $\mathcal{L} = [\texttt{abcdjkln}, \texttt{aefjkmn}, \texttt{abgchdjkln}, \ldots]$. Process discovery techniques are able to discover process models such as the Petri net shown in Fig. 2. Conformance deals with comparing an *a priori* process model with the observed behavior as recorded in the log and aims at detecting inconsistencies/deviations between a process model and its corresponding execution log. In other words, it checks for any violation

case id	event id	properties				
		time stamp	activity	resource	cost	...
1	10001001	01-10-2010:09:32	register	Bob	10	...
	10001002	02-10-2010:11:17	high insurance check	Alice	15	...
	10001003	02-10-2010:16:43	high medical history check	Alice	15	...
	10001004	03-10-2010:13:54	contact hospital	Alice	20	...
	10001005	04-10-2010:18:32	decide	Wil	150	...
	10001006	05-10-2010:09:05	prepare notification	Bob	10	...
	10001007	05-10-2010:10:13	send notification by email	Bob	10	...
	10001008	05-10-2010:10:44	archive	Bob	10	...
2	10002001	01-10-2010:11:01	register	Anita	10	...
	10002002	04-10-2010:14:23	low insurance check	Anu	12	...
	10002003	05-10-2010:10:37	low medical history check	Anu	12	...
	10002004	08-10-2010:08:16	decide	JC	50	...
	10002005	10-10-2010:14:05	prepare notification	Anita	10	...
	10002006	11-10-2010:15:13	send notification by post	Anita	10	...
	10002007	14-10-2010:9:41	archive	Anita	10	...
3	10001231	11-11-2010:10:32	register	Mike	11	...
	10001232	12-11-2010:12:13	high insurance check	Kate	17	...
	10001233	12-11-2010:13:14	send questionnaire	Mike	23	...
	10001234	22-11-2010:14:23	high medical history check	Kate	17	...
	10001235	22-11-2010:15:41	receive response	Sara	17	...
	10001236	03-12-2010:06:54	contact hospital	Peter	23	...
	10001237	04-12-2010:08:12	decide	Chase	100	...
	10001238	15-12-2010:13:05	prepare notification	Sasha	10	...
	10001239	15-12-2010:17:13	send notification by email	Sasha	10	...
	10001240	17-12-2010:18:14	archive	Mike	10	...
4	10006711	03-01-2011:13:01	register	Tom	10	...
	10006712	04-01-2011:11:33	send questionnaire	Harry	12	...
	10006713	06-01-2011:09:43	low insurance check	Mika	12	...
	10006714	15-01-2011:11:17	low medical history check	Mika	12	...
	10006715	18-01-2011:19:16	decide	Mark	80	...
	10006716	20-01-2011:17:05	prepare notification	Dash	10	...
	10006717	31-01-2011:05:13	send notification by post	Kim	10	...
	10006718	04-02-2011:17:23	skip response	Tom	12	...
	10006719	14-02-2011:09:41	archive	Kate	10	...
...	...	...	...	...	...	...

`a` - register
`b`-high insurance check
`c`-high medical history check
`d`-contact hospital
`e`-low insurance check
`f`-low medical history check
`g`-send questionnaire
`h`-receive response
`i`-skip response
`j`-decide
`k`-prepare notification
`l`-send notification by email
`m`-send notification by post
`n`-archive

project

```
a b c d j k l n
a e f j k m n
a b g c h d j k l n
a g e f j k m i n
...
```

learn

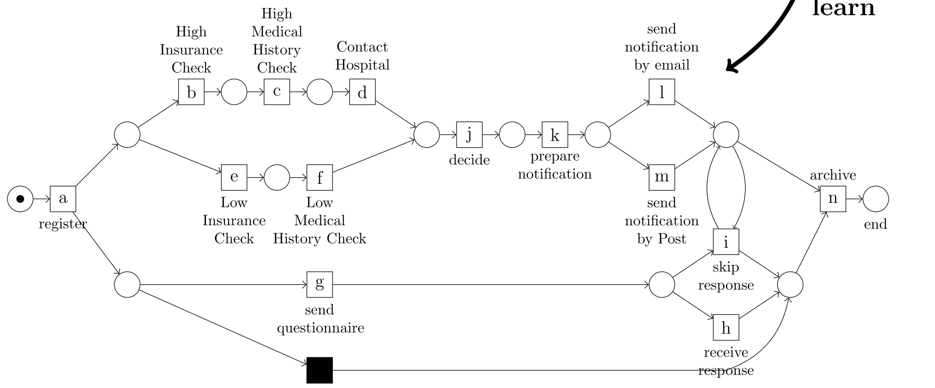


Fig. 2. Process discovery aims to learn a process model (in this case a Petri net) from event logs. An event log consists of events related to cases and referring to activities. To discover control flow, traces are projected onto activity names.

between *what was expected to happen* and *what actually has happened*. Enhancement deals with extending or improving an existing model based on information about the process execution in an event log. For example, annotating a process model with performance data to show bottlenecks, throughput times, and so on.

Being a relatively young research discipline, several process mining challenges remain to be addressed. The process mining manifesto [9] lists 11 challenges. The fourth challenge is dealing with concept drift and, thus far, a little work has been done on this highly relevant topic [10], [11].

B. Concept Drift

Concept drift [12] in machine learning and data mining refers to situations when the relation between the input data and the target variable, which the model is trying to predict, changes over time in unforeseen ways. Therefore, the accuracy of the predictions may degrade over time. To prevent that, predictive models need to be able to adapt online, i.e., to update themselves regularly with new data. The setting is typically looped over an infinite data stream as follows: 1) receive new data; 2) make a prediction; 3) receive feedback (the true target value); and 4) update the predictive model. While operating under such circumstances, predictive models are required: 1) to react to concept drift (and adapt if needed) as soon as possible; 2) to distinguish drifts from once-off noise and adapt to changes, but be robust to noise; and 3) to operate in less than data arrival time and use limited memory for storage. In this setting, many adaptive algorithms have been developed (e.g., overviews [13], [14]).

Concept drift is a relatively young research topic that has gained popularity in data mining and machine learning communities in the last 10 years. Concept drift research primarily has been focusing on two directions: 1) how to detect drifts (changes) online (e.g., [15]–[20]) and 2) how to keep predictive models up to date (e.g., [21]–[23]). Concept drift has been shown to be important in many applications (e.g., [24]–[26]). The basis for drift detection could be a raw data stream, a stream of prediction errors, and, more rarely, a stream of predictions or a stream of updated model parameters. Two types of concept drift detection approaches have been used: monitoring evolution of a stream [15], [17] or comparing data distributions in two time windows [16], [18]. The cumulative sum (CUSUM) approach [27] is a representative sequential analysis technique for change detection, different extensions to which have been proposed. One notable example is computational intelligence-based CUSUM or CI-CUSUM [19] that aims to detect a nonstationarity condition by monitoring a multidimensional vector, i.e., multiple features. Adaptive windowing [16] is a representative approach for online change detection using an adaptive size sliding detection window. In this paper, we consider offline change detection and its localization and therefore focus on studying what features to monitor and how to identify when these characteristics change.

III. Related Work

Over the last two decades many researchers have been working on process flexibility, e.g., making workflow systems adaptive. In [28] and [29] collections of typical change patterns are described. In [30] and [31] extensive taxonomies of the various flexibility approaches and mechanisms are provided. Ploesser *et al.* [32] have classified business process changes into three broad categories: 1) sudden; 2) anticipatory; and 3) evolutionary. This classification is used in this paper, but now in the context of event logs.

Despite the many publications on flexibility, most process mining techniques assume a process to be in a steady state. A notable exception is the approach in [33]. This approach uses process mining to provide an aggregated overview of all changes that have happened so far. This approach, however, assumes that change logs are available, i.e., modifications of the workflow model are recorded. At this point of time, very few information systems provide such change logs. Therefore, this paper focuses on concept drift in process mining assuming only an event log as input.

The topic of concept drift is well studied in various branches of the data mining and machine learning community. Concept drift has been studied in both supervised and unsupervised settings and has been shown to be important in many applications [12], [14], [25], [26], [34]–[37]. The problem of concept drift, however, has not been studied in the process mining setting. Unlike in data mining and machine learning, where concept drift focuses on changes in simple structures such as variables, concept drift in process mining deals with changes to complex artifacts such as process models describing concurrency, choices, loops, and cancelation. Although experiences from data mining and machine learning can be used to investigate concept drift in process mining, the complexity of process models and the nature of process change pose new challenges. This paper extends the work presented in [10]. In this extended paper, we introduce the topic of concept drift in process mining and present the basic idea and the features capturing the characteristics of traces in an event log in a more rigorous manner. In addition, this extended paper provides a generic framework for handling concept drifts in process mining and presents details on the realization of the approach in the ProM framework. Furthermore, this paper reports new experimental results of the proposed approach. More specifically, in this extended paper, we study the influence of population size on change point detection and the applicability of the approach in dealing with gradual drifts. In addition, we present the results of applying the approach on a real-life case study from a large Dutch municipality.

Recently, Carmona and Gavaldà [11] have proposed an online technique for detecting process changes. They first created an abstract representation of the process in the form of polyhedra using the prefixes of some initial traces in the event log. Subsequent traces are sampled and assessed whether they lie within the polyhedra or not. If a sample lies within the polyhedra, it is considered to be from the same process. If significant number of samples lies outside the polyhedra, a process change is said to be detected. This work differs from our approach in several ways: 1) this approach constructs an abstract representation of a process unlike ours where we consider features characterizing the traces and 2) this technique is applicable only for change detection whereas our framework is applicable for both change (point) detection and change localization. Furthermore, the tool support provided by the authors has some limitations in its applicability. The tool does not detect change points and does not work on logs with multiple process changes, i.e., it does not detect the presence/absence of multiple changes and does not report when (the trace index) process changes have happened. The tool just reports that a change exists and terminates (if changes exist) and does not terminate if no changes exist. In contrast, our tool can handle multiple process changes and can detect both the presence of and the points of change in addition to being able to assist in change localization.

IV. Characterization of Changes in Business Processes

In this section, we discuss the various aspects of process change. Initially, we describe change perspectives (control flow, data, and resource). Then, the different types of drift (sudden, gradual, recurring, periodic, and incremental) are discussed.

A. Perspectives of Change

There are three important perspectives in the context of business processes: 1) control flow; 2) data; and 3) resource. One or more of these perspectives may change over time.

1) *Control flow/behavioral perspective:* This class of changes deals with the behavioral and structural changes in a process model. Just like the design patterns in

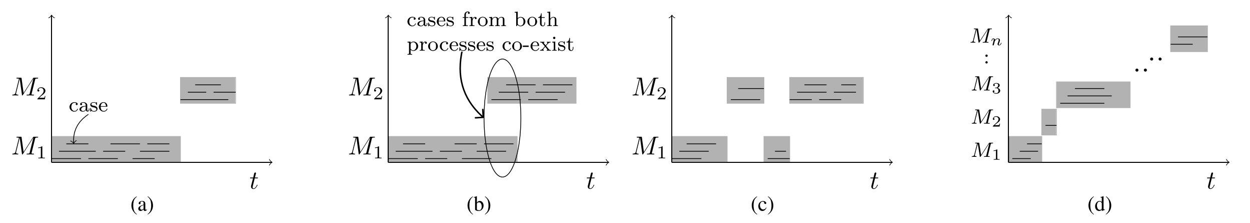


Fig. 3. Different types of drifts. x-axis: time. y-axis: process variants. Shaded rectangles: process instances. (a) Sudden drift. (b) Gradual drift. (c) Recurring drift. (d) Incremental drift.

software engineering, there exist change patterns capturing the common control-flow changes [29]. Control-flow changes can be classified into operations such as insertion, deletion, substitution, and reordering of process fragments. For example, an organization which used to collect a fee after processing and acceptance of an application can now change their process to enforce payment of that fee before processing an application. Here, the reordering change pattern had been applied on the payment and the application processing process fragments. As another example, with the addition of new product offerings, a choice construct is inserted into the product development process of an organization. In the context of PAISs, various control-flow change patterns have been proposed in [28], [29]. Most of these control-flow change patterns are applicable to traditional information/workflow systems as well.

Sometimes, the control-flow structure of a process model can remain intact but the behavioral aspects of a model change. For example, consider an insurance agency that classifies claims as high or low depending on the amount claimed. An insurance claim of €1000 which would have been classified as high last year is categorized as a low insurance claim this year because of the organization's decision to increase the claim limit. The structure of the process remains intact but the routing of cases changes.

2) *Data perspective:* This class of changes refer to the changes in the production and consumption of data and the effect of data on the routing of cases. For example, it may no longer be required to have a particular document when approving a claim.
3) *Resource perspective:* This class deals with the changes in resources, their roles, and organizational structure, and their influence on the execution of a process. For example, there could have been a change pertaining to who executes an activity. Roles may change and people may change roles. As another example, certain execution paths in a process could be enabled (disabled) upon the availability (nonavailability) of resources. Furthermore, resources tend to work in a particular manner and such working patterns may change over time, e.g., a resource can have a tendency of executing a set of parallel activities in a specific sequential order. Such working patterns could be more prominent when only few resources are available; the addition of new resources can remove this bias.

B. Nature of Drifts

With the duration for which a change is active, we can classify changes into *momentary* and *permanent*. Momentary changes are short lived and affect only a very few cases, whereas permanent changes are persistent and stay for a while [31]. In this paper, we focus on permanent changes as momentary changes often cannot be discovered because of insufficient data.[2] Momentary changes correspond to the notion of outliers/noise in data mining. Changes are perceived to induce a drift in the concept (process behavior). As shown in Fig. 3, we identify four classes of drifts.

1) *Sudden drift:* This corresponds to a substitution of an existing process M_1 with a new process M_2, as shown in Fig. 3(a). M_1 ceases to exist from the moment of substitution. In other words, all cases (process instances) from the instant of substitution emanate from M_2. This class of drifts is typically seen in scenarios such as emergencies, crisis situations, and change of law. As an example, a new regulation by the finance ministry of India mandates all banks to procure and report the customer's personal account number in their transactions.
2) *Gradual drift:* This refers to the scenario, as shown in Fig. 3(b) where a current process M_1 is replaced with a new process M_2. Unlike the sudden drift, here both processes coexist for some time with M_1 discontinued gradually. For example, a supply chain organization might introduce a new delivery process. This process is, however, applicable only for orders taken henceforth. All previous orders still have to follow the former delivery process.
3) *Recurring drift:* This corresponds to the scenario where a set of processes reappear after some time (substituted back and forth), as shown in Fig. 3(c). It is quite natural to observe such a phenomenon with processes having a seasonal influence. For example, a travel agency might deploy a different process to attract customers during Christmas period. The recurrence of processes may be periodic or nonperiodic. An example of a nonperiodic recurrence is the deployment of a process subjected to market conditions. The point of deployment and the duration of deployment are both dependent on external factors (here, the market conditions). Periodic drifts may be caused by seasonal effects,

[2]To analyze momentary changes we can also use standard conformance checking techniques to discover deviations from some normative model [38].

e.g., during the summer holidays there tends to be less demand and fewer resources thus influencing the process.

4) *Incremental drift:* This refers to the scenario where a substitution of process M_1 with M_N is done via smaller incremental changes, as shown in Fig. 3(d). This class of drifts is more pronounced in organizations adopting an agile BPM methodology and in processes undergoing sequences of quality improvements (most total quality management) initiatives are examples of incremental change [39]).

Recurring and incremental drifts in Fig. 3 are shown as discrete sudden changes. These two types of concept drift, however, can also be gradual. Similar categorization of drifts have been proposed in [40] in the context of machine learning. Drifts in [40] are further classified based on the severity of change into severe (and intersected). The categories of severity, as defined in [40], are too coarse to be applied to business process changes. Nonetheless, the degree of severity in process changes and their impact on dealing with concept drifts is an interesting topic for further research. In the rest, we propose approaches to detect potential control-flow changes in a process manifested as sudden/gradual drifts over a period. Detecting drifts in the other perspectives are beyond the scope of this paper. In addition, as already shown in Fig. 1, we focus on offline concept drift analysis (although our techniques can easily be adapted to the online setting). In practice, a mixture of any or all of the drifts may happen.

V. Basic Idea of Drift Detection in Event Logs

In this section, we present the basic idea for the detection of changes by analyzing event logs. Initially, we introduce the notations used in this paper.

1) $\mathcal{A}$ is the set of activities. $\mathcal{A}^+$ is the set of all nonempty finite sequences of activities from $\mathcal{A}$.
2) A process instance (i.e., case) is described as a trace over $\mathcal{A}$, i.e., a finite sequence of activities. Examples of traces are abcd and abbbad.
3) Let $\mathbf{t} = \mathbf{t}(1)\mathbf{t}(2)\mathbf{t}(3)\ldots\mathbf{t}(n) \in \mathcal{A}^+$ be a trace over $\mathcal{A}$. $|\mathbf{t}| = n$ is the length of the trace $\mathbf{t}$. $\mathbf{t}(k)$ is the k^{th} activity in the trace and $\mathbf{t}(i, j)$ is the continuous subsequence of $\mathbf{t}$ that starts at position i and ends at position j. $\mathbf{t}^i = \mathbf{t}(i, |\mathbf{t}|)$ represents the suffix of $\mathbf{t}$ that begins at position i.
4) An event log, $\mathcal{L}$, corresponds to a multiset (or bag) of traces from $\mathcal{A}^+$. For example, $\mathcal{L} = [\text{abcd, abcd, abbbad}]$ is a log consisting of three cases. Two cases follow trace abcd and one case follows trace abbbad.
5) $\mathbb{N}, \mathbb{N}_0$, and $\mathbb{R}_0^+$ are the set of all natural numbers, the set of all natural numbers including zero, and the set of all positive real numbers including zero, respectively.

We can consider an event log $\mathcal{L}$ as a time series of traces (traces ordered based on the timestamp of the first event). Fig. 4 shows such a perspective on an event log along with change points in the sudden drift scenario. The basic premise in handling concept drifts is that the characteristics of the traces before the change point differ from the characteristics of the traces after the change point. The problem of change point detection is then to identify the points in time where the process has changed, if any. Change point detection involves two primary steps:

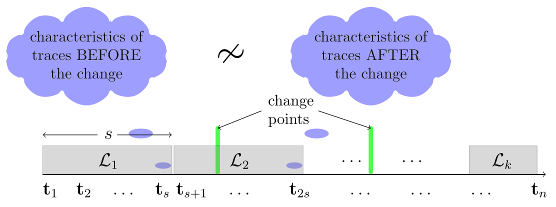

characteristics of traces BEFORE the change
$\not\approx$
characteristics of traces AFTER the change
change points
s
$\mathcal{L}_1$
$\mathcal{L}_2$
$\mathcal{L}_k$
$\mathbf{t}_1$ $\mathbf{t}_2$ … $\mathbf{t}_s$ $\mathbf{t}_{s+1}$ … $\mathbf{t}_{2s}$ … … … $\mathbf{t}_n$

Fig. 4. Event log visualized as a time series of traces along with change points. The basic premise of change (point) detection is that characteristic differences exist in the traces before and after the change.

1) capturing the characteristics of the traces;
2) identifying when the characteristics change.

We refer to the former step as feature extraction and the latter step as drift detection. The characteristics of the traces can either be defined for each trace separately or can be done at a sublog level. An event log can be split into sublogs of s traces ($s \in \mathbb{N}$ is the split size). We can consider either overlapping or nonoverlapping sliding windows when creating such sublogs. Fig. 4 shows the scenario where two subsequent sublogs do not overlap. In this case, we have $k = \lceil \frac{n}{s} \rceil$ sublogs for an event log of n traces. Thus, the logs processed to determine the characteristics of traces can be observed as a data stream of feature values where statistical tests can be used to detect changes.

As mentioned earlier, dealing with concept drifts in process mining involves two primary steps. First, we need to capture the characteristics of traces; we propose a few feature sets that address this in Section VI. Second, we need to identify when these characteristics change; we look at techniques that address this in Section VII.

VI. Feature Extraction

Event logs are characterized by the relationships between activities. Dependencies between activities in an event log can be captured and expressed using the follows (or precedes) relationship, also referred to as causal footprints. For any pair of activities a, b $\in \mathcal{A}$, and a trace $\mathbf{t} = \mathbf{t}(1)\mathbf{t}(2)\mathbf{t}(3)\ldots\mathbf{t}(n) \in \mathcal{A}^+$, we say b follows a if and only if for all $1 \leq i \leq n$ such that $\mathbf{t}(i) = \text{a}$ there exists a j such that $i < j \leq n$ and $\mathbf{t}(j) = \text{b}$. In temporal logic notation: $\Box(\text{a} \Rightarrow (\Diamond\text{b}))$. We say a precedes b if and only if for all $1 \leq j \leq n$ such that $\mathbf{t}(j) = \text{b}$ there exists an i such that $1 \leq i < j$ and $\mathbf{t}(i) = \text{a}$, i.e., $\neg\text{a}\mathcal{W}\text{b}$ where $\mathcal{W}$ is the *weak until* in linear temporal logic notation. The follows and precedes relationships can be lifted from traces to logs. If b follows a in all the traces in an event log, then we say that b *always follows* a. If b follows a only in some subset of the traces, then we say that b *sometimes follows* a. If b does not follow a in all traces, then we say that b *never follows* a. Consider an event log $\mathcal{L} = [\text{acaebfh, ahijebd, aeghijk}]$ containing three traces defined over $\mathcal{A} = \{\text{a, b, c, d, e, f, g, h, i, j, k}\}$. The following relations hold in $\mathcal{L}$: e always follows a, e never follows b, and b sometimes follows a. Fig. 5(a) shows the relationship between

	a	b	c	d	e	f	g	h	i	j	k	$f^{\mathcal{L}}_{\text{RC}}$	$f^{\mathcal{L}}_{\text{RE}}$
a	S	S	S	S	A	S	S	A	S	S	S	$\langle 2,9,0\rangle$	0.684
b	N	N	N	S	N	S	N	S	N	N	N	$\langle 0,3,8\rangle$	0.845
c	A	A	N	N	A	A	N	A	N	N	N	$\langle 5,0,6\rangle$	0.994
d	N	N	N	N	N	N	N	N	N	N	N	$\langle 0,0,11\rangle$	0.000
e	N	S	N	S	N	S	S	S	S	S	S	$\langle 0,8,3\rangle$	0.845
f	N	N	N	N	N	N	N	A	N	N	N	$\langle 1,0,10\rangle$	0.440
g	N	N	N	N	N	N	N	A	A	A	A	$\langle 4,0,7\rangle$	0.946
h	N	S	N	S	S	N	N	N	S	S	S	$\langle 0,6,5\rangle$	0.994
i	N	S	N	S	S	N	N	N	N	A	S	$\langle 1,4,6\rangle$	1.322
j	N	S	N	S	S	N	N	N	N	N	S	$\langle 0,4,7\rangle$	0.946
k	N	N	N	N	N	N	N	N	N	N	N	$\langle 0,0,11\rangle$	0.000

(a) (b) (c)

Fig. 5. Feature extraction (a) causal footprint matrix for all activity pairs (b) relation type count (RC) and (c) relation entropy (RE) feature values. A: always follows, N: never follows, and S: sometimes follows.

every pair of activities in $\mathcal{A}$. The value in a cell (i, j) is either A, S, or N corresponding to the relation whether the activity represented by column j always, sometimes, or never follows the activity represented by row i, respectively.

The variants of precedes relation can be defined along similar lines. The follows/precedes relationship is rich enough to reveal many control-flow changes in a process. We exploit this relationship and define various features for change detection.

We distinguish between two classes of features: 1) global and 2) local features. Global features are defined over an event log, whereas local features can be defined at a trace level. With the follows (precedes) relation, we propose two global features: 1) relation type count (RC) and 2) relation entropy (RE), and two local features: 1) window count (WC) and 2) J measure. These features are defined as follows.

1) *RC:* The RC with respect to the follows (precedes) relation is a function, $f^{\mathcal{L}}_{\text{RC}}\colon \mathcal{A} \to \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0$, defined over the set of activities $\mathcal{A}$. $f^{\mathcal{L}}_{\text{RC}}$ of an activity, x $\in \mathcal{A}$, with respect to the follows (precedes) relation over an event log $\mathcal{L}$ is the triple $\langle c_A, c_S, c_N\rangle$ where c_A, c_S, and c_N are the number of activities in $\mathcal{A}$ that always, sometimes, and never follows (precedes) x, respectively, in the event log $\mathcal{L}$. For the event log $\mathcal{L}$ mentioned above, $f^{\mathcal{L}}_{\text{RC}}(\text{a}) = \langle 2, 9, 0\rangle$ because e and h always follows a while all other activities in $\mathcal{A} \setminus \{\text{e}, \text{h}\}$ sometimes follows a. $f^{\mathcal{L}}_{\text{RC}}(\text{i}) = \langle 1, 4, 6\rangle$ because only j always follows i; b, d, e, and k sometimes follows i while a, c, f, g, h, and i never follows i. Fig. 5(b) shows the RCs for all the activities in $\mathcal{A}$ [the value in a row corresponds to the RCs of the activity represented by that row in Fig. 5(a)].

For an event log containing $|\mathcal{A}|$ activities, this results in a feature vector of dimension $3 \times |\mathcal{A}|$ (if either the follows or the precedes relation is considered) or $2\times 3\times|\mathcal{A}|$ (if both the follows and the precedes relations are considered).

2) *RE:* The RE with respect to the follows (precedes) relation is a function, $f^{\mathcal{L}}_{\text{RE}}\colon \mathcal{A} \to \mathbb{R}^+_0$, defined over the set of activities. $f^{\mathcal{L}}_{\text{RE}}$ of an activity, x $\in \mathcal{A}$ with respect to the follows (precedes) relation is the entropy of the RC metric. In other words, $f^{\mathcal{L}}_{\text{RE}}(\text{x}) = -p_A \log_2(p_A) - p_S \log_2(p_S) - p_N \log_2(p_N)$ where $p_A = c_A/|\mathcal{A}|$, $p_S = c_S/|\mathcal{A}|$, and $p_N = c_N/|\mathcal{A}|$ and $\langle c_A, c_S, c_N\rangle = f^{\mathcal{L}}_{\text{RC}}(\text{x})$.

For the above example event log $\mathcal{L}$, $f^{\mathcal{L}}_{\text{RE}}(\text{a}) = 0.684$ (corresponding to $f^{\mathcal{L}}_{\text{RC}}(\text{a}) = \langle 2, 9, 0\rangle$) and $f^{\mathcal{L}}_{\text{RE}}(\text{i}) = 1.322$ (corresponding to $f^{\mathcal{L}}_{\text{RC}}(\text{i}) = \langle 1, 4, 6\rangle$). Fig. 5(c) shows the RE for all the activities in $\mathcal{A}$ [the value in a row corresponds to the RE of the activity represented by that row in Fig. 5(a)].

For an event log containing $|\mathcal{A}|$ activities, this results in a feature vector of dimension $|\mathcal{A}|$ or $2 \times |\mathcal{A}|$ depending on whether either or both of the follows/precedes relations are considered.

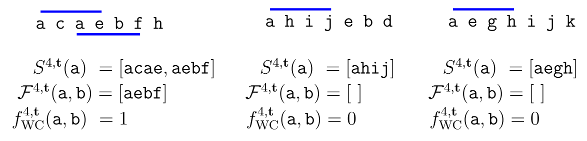

a c a e b f h
$S^{4,\mathbf{t}}(\text{a}) = [\text{acae}, \text{aebf}]$
$\mathcal{F}^{4,\mathbf{t}}(\text{a},\text{b}) = [\text{aebf}]$
$f^{4,\mathbf{t}}_{\text{WC}}(\text{a},\text{b}) = 1$
a h i j e b d
$S^{4,\mathbf{t}}(\text{a}) = [\text{ahij}]$
$\mathcal{F}^{4,\mathbf{t}}(\text{a},\text{b}) = [\ ]$
$f^{4,\mathbf{t}}_{\text{WC}}(\text{a},\text{b}) = 0$
a e g h i j k
$S^{4,\mathbf{t}}(\text{a}) = [\text{aegh}]$
$\mathcal{F}^{4,\mathbf{t}}(\text{a},\text{b}) = [\ ]$
$f^{4,\mathbf{t}}_{\text{WC}}(\text{a},\text{b}) = 0$

Fig. 6. WC values for the relation b follows a for the different traces in the event log.

3) *WC:* Given a window of size $l \in \mathbb{N}$, the WC with respect to follows (precedes) relation is a function, $f^{l,\mathbf{t}}_{\text{WC}}\colon \mathcal{A}\times\mathcal{A} \to \mathbb{N}_0$, defined over the set of activity pairs. Given a trace $\mathbf{t}$ and a window of size l, let $S^{l,\mathbf{t}}(\text{a})$ be the bag of all subsequences $\mathbf{t}(i, i + l - 1)$, such that $\mathbf{t}(i) = \text{a}$.[3] Let $\mathcal{F}^{l,\mathbf{t}}(\text{a}, \text{b}) = [\mathbf{s} \in S^{l,\mathbf{t}}(\text{a}) \mid \exists_{1<k\leq|\mathbf{s}|}\ \mathbf{s}(k) = \text{b}]$, i.e., the bag of subsequences in $\mathbf{t}$ starting with a and followed by b within a window of length l. The WC of the relation b follows a, $f^{l,\mathbf{t}}_{\text{WC}}(\text{a}, \text{b}) = |\mathcal{F}^{l,\mathbf{t}}(\text{a}, \text{b})|$.

Fig. 6 shows the WC values for the relation b follows a in the event log $\mathcal{L}$ using a window of length four.

4) *J measure:* Smyth and Goodman [41] have proposed a metric called J measure based on [42] to quantify the information content (goodness) of a rule. We adopt this metric as a feature to characterize the significance of relationship between activities. The basis lies in the fact that we can consider the relation b follows a as a rule: if activity a occurs, then activity b will probably occur. The J measure with respect to follows (precedes) relation is a function $f^{l,\mathbf{t}}_{\text{J}}\colon \mathcal{A} \times \mathcal{A} \to \mathbb{R}^+$ defined over the set of activity pairs and a given window of length $l \in \mathbb{N}$. Let $p^{\mathbf{t}}(\text{a})$ and $p^{\mathbf{t}}(\text{b})$ are the probabilities of occurrence of activities a and b, respectively, in a trace $\mathbf{t}$. Let $p^{l,\mathbf{t}}(\text{a}, \text{b})$ be the probability that b follows a within a window of length l, i.e., $p^{l,\mathbf{t}}(\text{a}, \text{b}) = |\mathcal{F}^{l,\mathbf{t}}(\text{a}, \text{b})|/|S^{l,\mathbf{t}}(\text{a})|$. Then, the J measure for a window of length l is defined as $f^{l,\mathbf{t}}_{\text{J}}(\text{a}, \text{b}) = p^{\mathbf{t}}(\text{a})\text{CE}^{l,\mathbf{t}}(\text{a}, \text{b})$ where $\text{CE}^{l,\mathbf{t}}(\text{a}, \text{b})$ is the cross entropy of a and b (b follows a within a window of length l) and is defined as[4]

$$\text{CE}^{l,\mathbf{t}}(\text{a}, \text{b}) = p^{l,\mathbf{t}}(\text{a}, \text{b}) \log_2\left(\frac{p^{l,\mathbf{t}}(\text{a}, \text{b})}{p^{\mathbf{t}}(\text{b})}\right)$$

[3]If $i + l - 1 > |\mathbf{t}|$, then $\mathbf{t}(i, i + l - 1) = \mathbf{t}^i$, i.e., the suffix of the trace $\mathbf{t}$ starting at i.

[4]$\log_2(0/x)$ and $\log_2(x/0)$ for any $x \in \mathbb{R}^+_0$ is taken as 0.

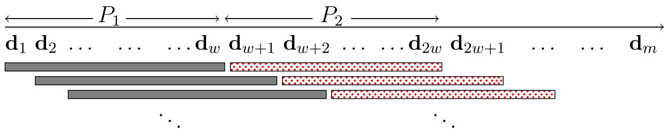

P_1
P_2
$\mathbf{d}_1\ \mathbf{d}_2 \dots \dots \dots \mathbf{d}_w\ \mathbf{d}_{w+1}\ \mathbf{d}_{w+2} \dots \dots \mathbf{d}_{2w}\ \mathbf{d}_{2w+1} \dots \dots\ \mathbf{d}_m$

Fig. 7. Basic idea of detecting drifts using hypothesis tests. The dataset of feature values is considered as a time series for hypothesis tests. P_1 and P_2 are two populations of size w.

$$+ (1 - p^{l,\mathbf{t}}(\mathrm{a}, \mathrm{b})) \log_2 \left(\frac{1 - p^{l,\mathbf{t}}(\mathrm{a}, \mathrm{b})}{1 - p^{\mathbf{t}}(\mathrm{b})} \right).$$

The J measure of a relation, b follows a, captures the dissimilarity between the *a priori* and *a posteriori* beliefs about b. In other words, it measures the difference between the *a priori* distribution of b (i.e., probability that b occurs in a trace and the probability that b does not occur), and the posteriori distribution of b (i.e., probability that b occurs in a trace given that a occurred and the probability that b does not occur in a trace given that a occurred).

The J measures for the relation b follows a using a window of length four for the three traces in the event log $\mathcal{L}$ in our previous example are 0.147, 0.032, and 0, respectively.

Normally, the window size is chosen to be the average trace length, i.e., the average number of events in a process instance, if no *a priori* information about the process is known. In case, we have some *a priori* information about the process, we can use the process characteristics to choose an appropriate window size. Having defined the features, we next look at the second step in change point detection, i.e., drift detection.

VII. Hypothesis Tests for Drift Detection

An event log can be transformed into a data stream/sequence $\mathcal{D}$ by choosing one of the feature sets defined in the previous section. The dataset $\mathcal{D}$ of feature values can be considered as a time series of m values, as shown in Fig. 7. Each $\mathbf{d}_i \in \mathcal{D}$ corresponds to the feature value(s) for a trace (or sublog) and can be a scalar or a vector (depending on the choice of feature).[5] Comparing with Fig. 4, $m = n$ or $m = k$ depending on whether the feature values are computed for each trace or for each sublog, respectively. As mentioned earlier, we expect a characteristic difference in the manifestation of feature values in the traces (sublogs) before and after the change points with the difference being more pronounced at the boundaries. To detect this, we can consider a series of successive populations of values (of size w) and investigate if there is a significant difference between two subsequent populations. The premise is that differences are expected to be perceived at change points provided appropriate characteristics of the change are captured as features. A moving window of size w is used to generate the populations. Fig. 7 shows a scenario where two populations $P_1 = \langle \mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_w \rangle$ and $P_2 = \langle \mathbf{d}_{w+1}, \mathbf{d}_{w+2}, \dots, \mathbf{d}_{2w} \rangle$ of size w are considered. In the next iteration, the populations correspond to $P_1 = \langle \mathbf{d}_2, \mathbf{d}_3, \dots, \mathbf{d}_{w+1} \rangle$ and $P_2 = \langle \mathbf{d}_{w+2}, \mathbf{d}_{w+3}, \dots, \mathbf{d}_{2w+1} \rangle$. Given a dataset of m values, the number of population pairs (iterations) will be $m - 2w + 1$.

We propose the use of statistical hypothesis testing to discover these change points. Hypothesis testing is a procedure in which a hypothesis is evaluated on a sample data. One of the important uses of hypothesis testing is to evaluate and compare groups of data. Numerous varieties of hypothesis tests exist [43]. The choice of a particular test is largely dependent on the nature of the data and the objectives of an experiment. For example, hypothesis tests can be classified into parametric and nonparametric tests. Parametric tests assume that the data have a particular distribution, e.g., normal, whereas the nonparametric tests do not make any assumption with regards to the data distribution. Because we do not know the *a priori* distribution of the feature values in an event log, we consider only nonparametric tests. Another perspective of classification is based on the number of samples (populations) on which the hypothesis is defined. We can classify the hypothesis tests into 1) one-sample; 2) two-sample; and 3) multisample tests. Because we need to analyze two populations for detecting drifts, we are interested in two-sample hypothesis tests. Another classification of hypothesis tests is concerned with the dimensionality of each data element in a sample. Tests dealing with scalar data elements are called univariate tests while those dealing with vector data elements are called multivariate tests. If only a particular activity or activity pair is considered, then every data item $\mathbf{d}_i \in \mathcal{D}$ is a scalar value corresponding to the trace/sublog i. If we, however, consider sets of activities or activity pairs, then each data item is a vector. Therefore, we need to consider both univariate and multivariate hypothesis tests.

We will use the univariate two-sample Kolmogorov–Smirnov test (KS test) and Mann–Whitney U test (MW test) as hypothesis tests for univariate data, and the two sample Hotelling T^2 test for multivariate data. The KS test evaluates the hypothesis "Do the two independent samples represent two different cumulative frequency distributions?" whereas the MW test evaluates the hypothesis "Do the two independent samples have different distributions with respect to the rank ordering of the values?". The multivariate Hotelling T^2 test is a generalization of the t-test and evaluates the hypothesis "Do the two samples have the same mean pattern?". All of these tests yield a significance probability assessing the validity of the hypothesis on the samples. We refer [43] for a classic introduction to various hypothesis tests.

VIII. Framework

We propose the framework shown in Fig. 8 for analyzing concept drifts in process mining. The framework identifies the following steps:

1) *Feature extraction and selection:* This step pertains in defining the characteristics of the traces in an event log. In this paper, we have defined four features that characterize the control-flow perspective of process instances

[5]The RE, WC, and J measure feature sets proposed in Section VI generate univariate (scalar) and multivariate (vector) data depending on whether we consider an individual activity/activity pair or a set of activities/activity pairs, respectively. The RC feature set always generates multivariate data.

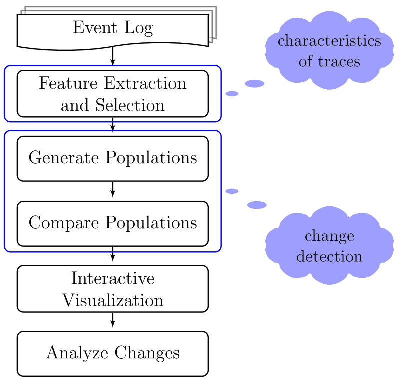

Event Log
Feature Extraction and Selection
Generate Populations
Compare Populations
Interactive Visualization
Analyze Changes
characteristics of traces
change detection

Fig. 8. Framework for handling concept drifts in process mining.

in an event log. Depending on the focus of analysis, we may define additional features, e.g., if we are interested in analyzing changes in organizational/resource perspective, we may consider features derived from social networks as a means of characterizing the event log. In addition to feature extraction, this step also involves feature selection. Feature selection is important when the number of features extracted is large. We may consider dimensionality reduction techniques [44], [45] such as PCA [46] or random projection [47] to deal with high dimensionality.

2) *Generate populations:* An event log can be transformed into a data stream based on the features selected in the previous step. This step deals with defining the sample populations for studying the changes in the characteristics of traces. Different criteria/scenarios may be considered for generating these populations from the data stream. In Section VII, we have considered nonoverlapping, continuous, and fixed-size windows for defining the populations. We may also consider, for example, noncontinuous windows (there is a gap between two populations), adaptive windows (windows can be of different lengths) [16], and so on, which are more appropriate for dealing with gradual and recurring drifts.

3) *Compare populations:* Once the sample populations are generated, the next step is to analyze these populations for any change in characteristics. In this paper, we advocate the use of statistical hypothesis tests for comparing populations. The null hypothesis in statistical tests states that distributions (or means, or standard deviations) of the two sample populations are equal. Depending on desired assumptions and the focus of analysis, different statistical tests can be used.

4) *Interactive visualization:* The results of comparative studies on the populations of trace characteristics can be intuitively presented to an analyst. For example, the significance probabilities of the hypothesis tests can be visualized as a drift plot. Troughs in such a drift plot signify a change in the significance probability thereby implying a change in the characteristics of traces.

5) *Analyze changes:* Visualization techniques such as the drift plot can assist in identifying the change points. Having identified that a change had taken place, this step deals with techniques that assist an analyst in characterizing and localizing the change and in discovering the change process.

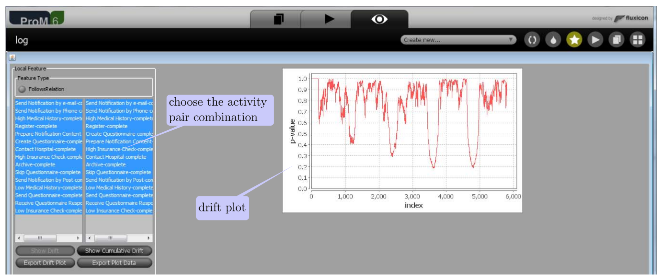

ProM 6
log
Local Feature
Feature Type
FollowsRelation
Send Notification by e-mail-co
Send Notification by Phone-c
High Medical History-complete
Register-complete
Prepare Notification Content-
Create Questionnaire-comple
Contact Hospital-complete
High Insurance Check-comple
Archive-complete
Skip Questionnaire-complete
Send Notification by Post-com
Low Medical History-complete
Send Questionnaire-complete
Receive Questionnaire Respo
Low Insurance Check-comple
Show Drift
Show Cumulative Drift
Export Drift Plot
Export Plot Data
choose the activity pair combination
drift plot
p-value
index
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
1,000
2,000
3,000
4,000
5,000
6,000

Fig. 9. Visualization of the drift plot in the concept drift plug-in in ProM.

The framework can be used for designing new change detection approaches.

IX. Implementation

The concepts presented in this paper have been realized as the concept drift plug-in in the ProM[6] framework. ProM is a plug-able environment for process mining envisioned to provide a common basis for all kinds of process mining techniques ranging from importing, exporting, and filtering event logs (process models) to analysis and visualization of results. Over years, ProM has emerged to be the de facto standard for process mining. The concept drift plug-in implements all of the steps in the proposed framework and can be easily extended with additional elements (e.g., new features can be easily added). The plug-in supports visualization of the significance probability for the hypothesis tests as a drift plot. Fig. 9 shows a drift plot from the plug-in.

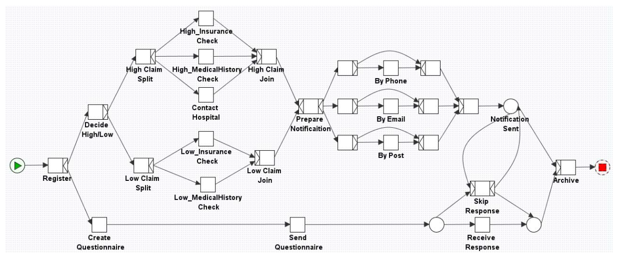
X. Experimental Results and Discussion

Now, we put the ideas proposed for handling concept drifts in practice. Initially, we illustrate the effectiveness of the proposed approaches using a synthetic example of an insurance claim process and later discuss the results from a real-life case study in a large Dutch municipality.

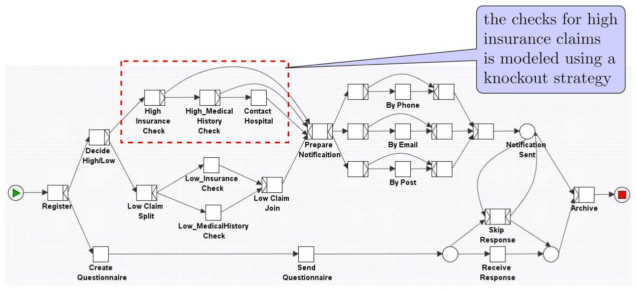
A. Synthetic Log-Insurance Claim Process

This process corresponds to the handling of health insurance claims in a travel agency. Upon registration of a claim, a general questionnaire is sent to the claimant. In parallel, a registered claim is classified as high or low. For low claims, two independent tasks: 1) check insurance and 2) check medical history need to be executed. For high claims, three tasks need to be executed: 1) check insurance; 2) check medical history; and 3) contact doctor/hospital for verification. If one of the checks shows that the claim is not valid, then the claim is rejected; otherwise, it is accepted. A cheque and acceptance decision letter is prepared in cases where a claim is accepted while a rejection decision letter is created for rejected claims. In both cases, a notification is sent to the claimant.

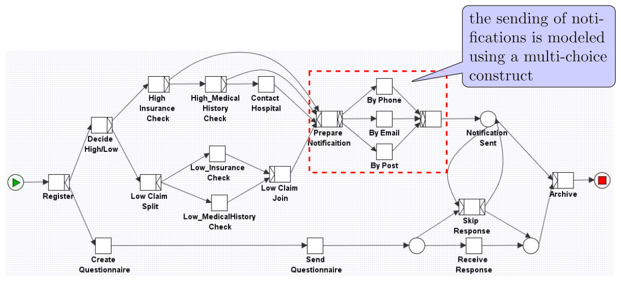
[6]See www.processmining.org for more information and to download ProM.



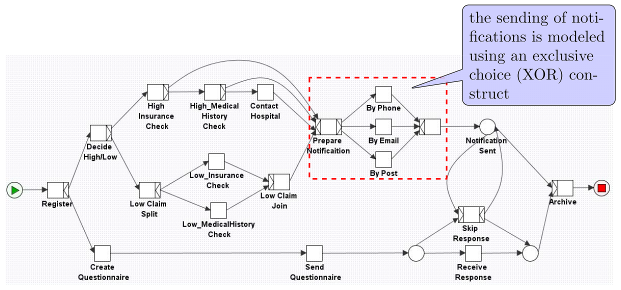
(a)



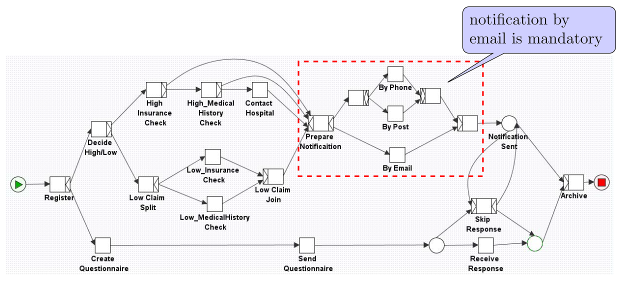
(b)



(c)



(d)



(e)

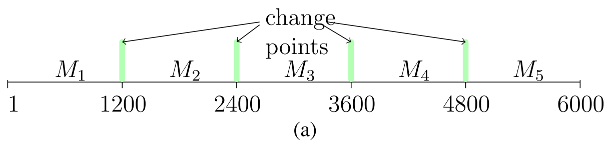
Fig. 10. Variants of an insurance claim process of a travel agency represented in YAWL notation. Dashed rectangles: regions of change from its previous model. (a) Model 1. (b) Model 2. (c) Model 3. (d) Model 4. (e) Model 5.

Three modes of notification are supported by: 1) email; 2) telephone (fax); and 3) postal mail. The case should be archived upon notifying the claimant. This can be done with or without the response for the questionnaire. The decision of ignoring the questionnaire, however, can only be made after a notification is sent. The case is closed upon completion of archiving task.

Fig. 10 shows five variants of this process represented in YAWL [48] notation. Dashed rectangles: a change has been done in the process model with respect to its previous variant. The changes can have various reasons. For example, in Fig. 10(a), the different checks for high insurance claims are modeled using a parallel (AND) construct. A claim, however, can be rejected if any one of the checks fail. In such cases, the time and resources spent on other checks go waste. To optimize this process, the agency can decide to enforce an order on these checks and proceed on checks only if the previous check results are positive. In other words, the process is modified with a *knockout* strategy [49] adopted for the process fragment involving the different checks for high insurance claims, as shown in Fig. 10(b). As another example, the OR-construct pertaining to the sending of notification to claimants in Fig. 10(c) has been modified to an exclusive-or (XOR) construct in Fig. 10(d). The organization could have taken a decision to reduce their workforce as a cost-cutting measure. Because of the availability of limited resources, they would like to minimize the redundancy of sending the notification through different modes of communication and restrict it to only one of the modes. Considering an event log containing cases that belong to such a mix of process variants, the objective of change point detection is to detect when the processes have changed. In this section, we illustrate the handling of concept drifts in the context of sudden and gradual drifts. We have modeled each of these five process variants in CPN tools [50] and simulated 1200 traces for each model.

1) Sudden Drift Change (Point) Detection: To simulate the sudden drift phenomenon, we created an event log $\mathcal{L}$ consisting of 6000 traces by juxtaposing each set of the 1200 traces. The event log contains 15 activities or event classes (i.e., $|\mathcal{A}| = 15$) and 58 783 events (which is the total number of events in the log for all the traces). Given this event log $\mathcal{L}$, our first objective is to detect the four change points pertaining to these five process variants, as shown in Fig. 11(a). Global features can be applied only at the log level; to facilitate this, we have split the log into 120 sublogs using a split size of 50 traces. In this scenario, the four change points corresponding to the five process variants are, as shown in Fig. 11(b). We have computed the follows RC of all 15 activities thereby generating a multivariate vector of 45 features for each sublog. We have applied the Hotelling T^2 hypothesis test on this multivariate dataset using a moving window population of size, $w = 10$. For this hypothesis test, we have randomly chosen 12 of the 45 features with a 10-fold cross validation.[7] Fig. 12(a) shows the average significance probability of the Hotelling T^2 test for the 10 folds on this feature set. The troughs in the plot signify that there is a change in the distribution of the feature

[7]The random selection of a subset of features is primarily for two reasons: 1) to deal with the curse of dimensionality and 2) the changes being centered around a few activities are prominently reflected only in those features corresponding to these activities.



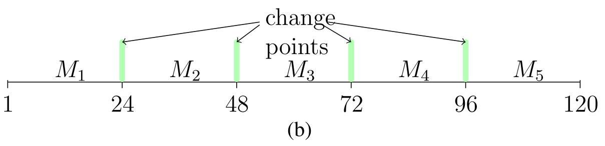
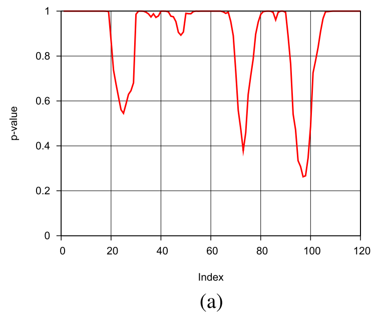
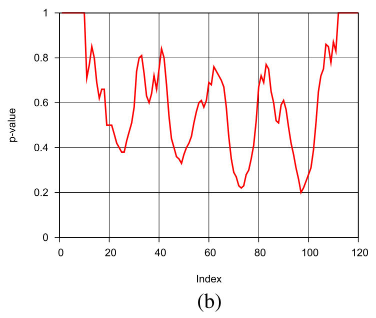


Fig. 11. Event log with traces from each of the five models juxtaposed. Also shown are change points between models both at the trace and sublog levels. The event log is split into 120 sublogs, each containing 50 traces. (a) Trace level. (b) Sub-log level.





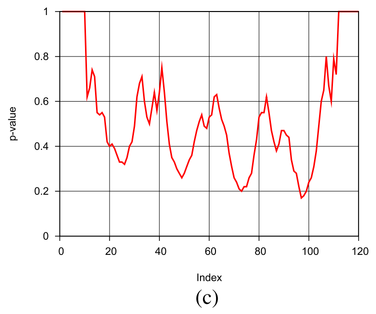
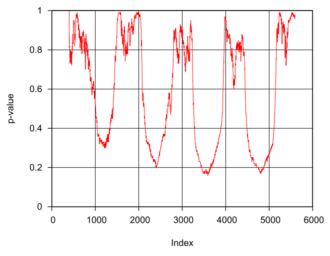


Fig. 12. (a) Significance probability of Hotelling T^2 test on relation counts. Average significance probability (over all activity pairs) of (b) KS test on J measure and (c) MW test on J measure. The event log is split into sublogs of 50 traces each. x-axis: sublog index. y-axis: significance probability of the test. Troughs: change points. Vertical grid lines: the actual (sudden) change points.

values in the log. In other words, they show that there is a drift (change) in the concept, which here corresponds to the process. It is interesting to observe that the troughs are observed around indexes 24, 72, and 96 which are indeed the points of change (remember that, we have split the log into 120 sublogs with the change points at indexes 24, 48, 72, and 96). The change at index 48 corresponding to the transition from M_2 to M_3 could not be uncovered using this feature set because the RCs would be alike for logs generated from these two process variants.

We have considered the J measure for each sublog and for every pair of activities, a and b in $\mathcal{A}$ (b follows a within a window of length $l = 10$). The univariate KS and the MW tests using a population of size $w = 10$ are applied on the J measure of each activity pair. Fig. 12(b) shows the average significance probability of the KS test on all activity pairs, whereas Fig. 12(c) shows the same for the MW test. We can observe that significant troughs are formed at indexes 24, 48, 72, and 96 which correspond to the actual change points. Unlike the RC feature, the J measure feature is able to capture all the four changes in the models. This can be attributed to the fact that the J measure uses the probability of occurrence of activities and their relations. In M_2, there could be cases where all the modes of notification are skipped (XOR construct). In M_3 at least one of the modes, however, needs to be executed (OR construct). This results in a difference in the distribution of activity probabilities and their relationship probabilities, which is elegantly captured by the J measure. Our experiences showed that KS test is more robust than the MW test. Henceforth, we report our results only using the KS test.

We have considered the J measure for each trace separately instead of at the sublog level. Each activity pair generates a vector of dimension 6000 corresponding to the J measure of that activity pair in each trace. The univariate KS test using a population size of $w = 400$ is applied to the vector



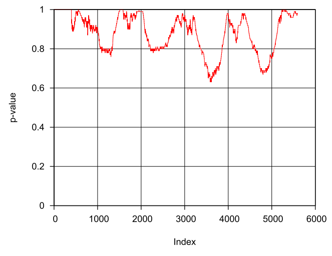


Fig. 13. Average significance probability (over all activity pairs) of KS test on the J measure and WC feature sets estimated for each trace. x-axis: trace index. y-axis: significance probability of the test. Troughs: change points. Vertical grid lines: actual (sudden) change points. (a) J-measure. (b) WC.

corresponding to each activity pair in $\mathcal{A} \times \mathcal{A}$. Fig. 13(a) shows the average significance probability of KS test on all activity pairs, whereas Fig. 13(b) shows the average significance probability of KS test on all activity pairs using the WC feature set. We can observe that significant troughs are formed at indexes 1200, 2400, 3600, and 4800. These are indeed the points where the models have been changed.

Influence of Population Size: It is imperative to note that the goodness of the results of hypothesis tests depends on the population size. The statistical analysis assumes that each population is independent. A good population size is largely dependent on the application and the focus of analysis. To study the influence of population size, we have considered the J measure for every pair of activities and the univariate KS test for change point detection. Fig. 14 shows the results for varying sizes of the population. We observe a lot of noise for small populations and the drift tends to be smooth as the population size increases. This can be attributed to the fact that as the population size increases (i.e., as we consider more cases), the variability in the nature of cases reduces and attains

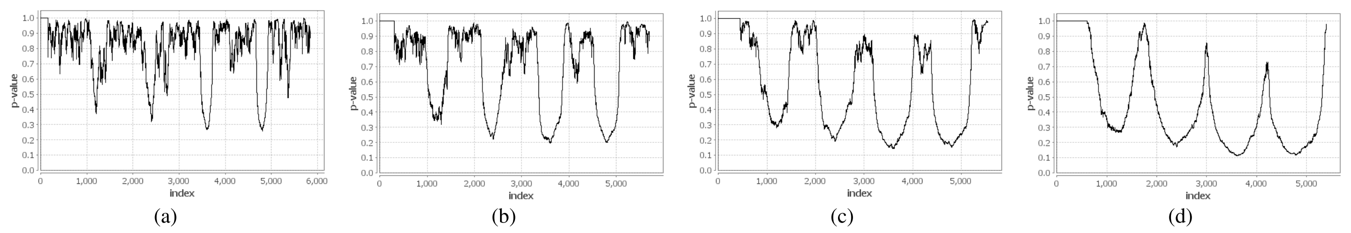


Fig. 14. Average significance probability (over all activity pairs) for different population sizes of KS test on the J measure estimated over all activity pairs in each trace. x-axis: trace index. y-axis: significance probability of the test. Troughs: change points. Vertical grid lines: actual (sudden) change points. (a) w = 150, (b) w = 300, (c) w = 450, and (d) w = 600.

a stability, e.g., there can be a flux of low-insurance claims initially and after a certain time the proportion stabilizes.

2) Sudden Drift Change Localization: Our second objective in handling concept drifts is that of change localization. To localize the changes (identify the regions of change), we need to consider activity pairs individually or subsets of activity pairs. For example, the change from M_1 to M_2 is localized in the region pertaining to high insurance claim checks. We expect characteristic changes in features pertaining to these activities and other activities related to these activities. For example, in M_1, the activities High Medical History Check and Contact Hospital always follow the activity Register whenever a claim is classified as high. In contrast, in M_2, these activities need not always follow Register because both these activities are skipped if High Insurance Check fails while Contact Hospital is skipped if High Medical History Check fails. During simulation, we have set the probability of success of a check to 90%. We have considered the WC feature for the activity relation Contact Hospital follows Register on a window length of $l = 10$ in each trace separately. Fig. 15(a) shows the significance probability of the univariate KS test using a population size of $w = 400$ on this feature. We can observe that one dominant trough is formed at index 1200 showing that there exists a change in the region between Register and Contact Hospital. No subsequent changes with respect to this activity pair can be noticed, which is indeed the case in the sequence of models used.

As another example, we have considered the activity Prepare Notification along with all the three Send Notification activities. There exists a change pertaining to these activities between models M_2 and M_3, M_3 and M_4, and M_4 and M_5. More specifically, we have considered the WC feature on the activity relations: Send Notification By Phone follows Prepare Notification, Send Notification By email follows Prepare Notification, and Send Notification By Post follows Prepare Notification. Fig. 15(b) shows the average significance probability of the univariate KS tests using a population size of $w = 400$ on the WC feature for various modes of send notification follows prepare notification. We observe three dominant troughs around indexes 2400, 3600, and 4800 signifying the changes in the models. Certain false alarms (minor troughs) can also be noticed in this plot. One means of alleviating this is to consider only those alarms with an average significance probability less than a threshold, δ. Another means is to consider a larger population size. In this fashion, by considering activities (and/or activity pairs) of interest, we can localize the regions of change. Furthermore, using this approach, we can obtain answers to diagnostic questions such as *Is there a change with respect to activity* a *in the process at time period* t?

The WC feature performs better in change localization in comparison with the J measure. This is because the J measure uses the probability of activities which can be affected because of changes anywhere in the process irrespective of our region of focus. For example, consider the J measure for the relation Contact Hospital follows Register. The probability of occurrence of both Register and Contact Hospital is affected by the changes in the process model corresponding to the sending of notifications as well, e.g., in M_3 because all the modes of send notification can be executed, the probability of Contact Hospital in a trace is smaller than a corresponding trace (Contact Hospital is executed) in M_4 where only one of the notifications is possible. Fig. 15(c) shows the significance probability of the univariate KS test on the J measure for the activity relation Contact Hospital follows Register, whereas Fig. 15(d) shows the average significance probability of the univariate KS tests on the J measure of various Send Notification modes following Prepare Notification using a population size of $w = 400$. Although the J measure can identify changes, it has problems localizing the change regions. Therefore, we recommend the use of WC feature for change localization.

3) Gradual Drift Change (Point) Detection: Now, we assess the accuracy of the proposed framework in handling gradual drifts. Recall that in gradual drifts, one concept fades gradually while the other takes over. This phenomenon of gradual change can be modeled in many ways. In this paper, we consider the scenario where the change is linear between two sources, as shown in Fig. 16(a). In this figure, we observe the fading of one concept M_1 and the taking over of another concept M_2 happen linearly. Within this setup, we can alter the extent to which the two concepts coexist. For the insurance claim example, we generated two event logs exhibiting gradual drifts by varying the duration of change. In the first case, the process variants M_1 and M_2 coexist between trace indexes 1000 and 1400, the variants M_2 and M_3 coexist between indexes 2200 and 2600, the variants M_3 and M_4 coexist between indexes 3400 and 3800, and the variants M_4 and M_5 coexist between indexes 4600 and 5000, as shown in Fig. 16(b). The point of cross over is still retained at indexes 1200, 2400, 3600, and 4800.

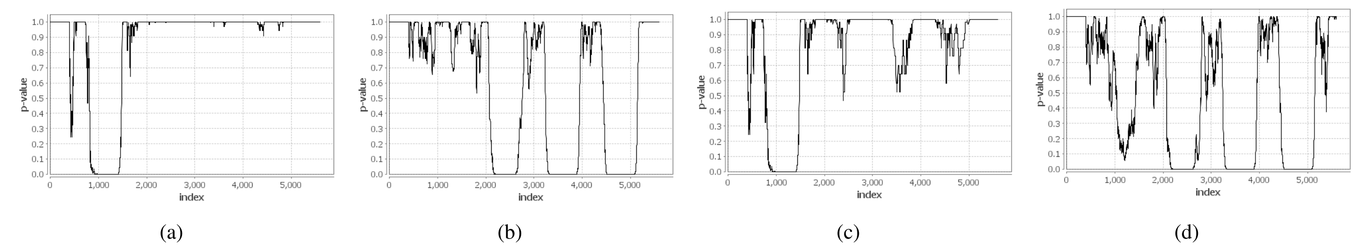


Fig. 15. (a) Significance probability of KS test for the relation, Contact Hospital follows Register using the WC feature. (b) Average significance probability (over activity pairs) of KS test estimated for the various modes of Send Notification follows Prepare Notification relation using the WC feature. (c) Significance probability of KS test for the relation, Contact Hospital follows Register using the J measure. (d) Average significance probability (over activity pairs) of KS test estimated for the various modes of Send Notification follows Prepare Notification relation using the J measure. Troughs: change point with respect to these activities. x-axis: trace index. y-axis: significance probability of the test.

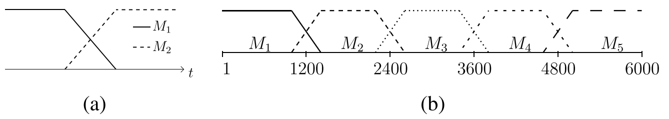


Fig. 16. Experimental setup for gradual drifts. (a) Generic scenario of linear gradual change between different process variants. (b) Linear gradual change with an overlapping window of 400 instances between any two process variants.

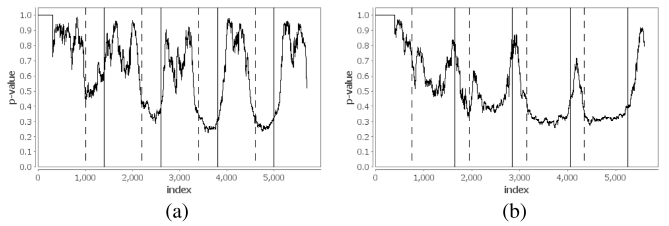


Fig. 17. Average significance probability (over all activity pairs) of KS test on the J measure for linear gradual change with an overlapping window of (a) 400 instances between any two process variants and (b) 900 instances between any two process variants. Dashed vertical grid lines: actual onset of gradual change. Corresponding solid vertical grid lines: actual end points of gradual change.

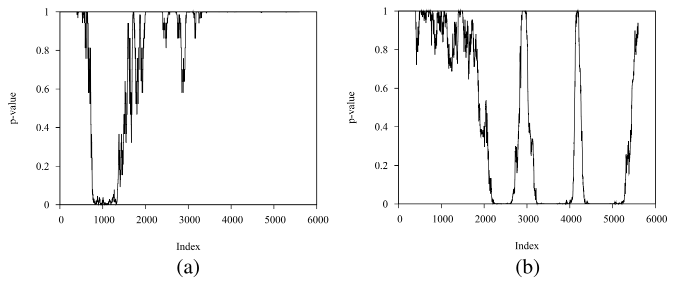


Fig. 18. (a) Significance probability of KS test for the relation, Contact Hospital follows Register using the WC feature. (b) Average significance probability (over activity pairs) of KS test estimated for the various modes of Send Notification follows Prepare Notification relation using the WC feature.

Fig. 17(a) shows the average significance probability of the univariate KS test over all activity pairs on the J measure using a population of size 300. We can observe that the proposed approach is able to detect the change points. It is, however, noteworthy that the width of the troughs is wider (at the top) in the gradual drift scenario when compared with the sudden drift scenario [compare Figs. 14(b) and 17(a)] signifying an earlier onset of change in the gradual drift phenomenon. We generated another event log with a linear gradual drift but with a longer duration of change. In this case, the process variants M_1 and M_2 coexist between trace indexes 750 and 1650, the variants M_2 and M_3 coexist between indexes 1950 and 2850, the variants M_3 and M_4 coexist between indexes 3150 and 4050, and the variants M_4 and M_5 coexist between indexes 4350 and 5150. Fig. 17(b) shows the average significance probability of the univariate KS test over all activity pairs on the J measure using a population of size 450. Even in this case, we can clearly identify the points and the duration of change is captured as a much wider trough when compared with the sudden drift scenario [compare Figs. 14(c) and 17(b)].

4) Gradual Drift Change Localization: Similar to sudden drift change localization, we have considered the WC feature for the activity relation Contact Hospital follows Register on a window length of $l = 10$ in each trace separately on the gradual drift log with a longer duration of graduality (i.e., a log with linear gradual change with an overlapping window of 900 instances between any two process variants.). Fig. 18(a) shows the significance probability of the univariate KS test using a population size of $w = 400$ on this feature. We can observe that one dominant trough is formed at index 1200 showing that there exists a change in the region between Register and Contact Hospital. No subsequent changes with respect to this activity pair can be noticed, which is indeed the case in the sequence of models used. Unlike the sudden drift scenario, the onset of change, however, happens much earlier [compare it with Fig. 15(a)]. Fig. 18(b) shows the average significance probability of the univariate KS tests using a population size of $w = 400$ on the WC feature for various modes of send notification follows prepare notification. Even here, we observe three dominant troughs around indexes 2400, 3600, and 4800 signifying the changes in the models with earlier onsets of change [compare it with Fig. 15(b)].

B. Real-Life Log: Advertisement Permit Process of a Dutch Municipality

The synthetic event data created through simulation allow us to compare the controlled (ground truth) changes with the detected changes. We, however, have also applied our concept drift analysis techniques to various real-life event logs. Here, we report on a case study where we analyzed concept drifts in processes within a large Dutch municipality. Municipalities are interested in obtaining insights into their processes, e.g., the way they are planned to be executed vis-a-vis the way they are actually executed. Recently, different municipalities in the Netherlands have evinced interest in comparing their processes and learning from each other (the interested reader is referred to the CoSeLog project [51] for further information). Their vision is to have a form of standardization through a centrally managed process management system [52]–[55]. When analyzing event logs, we need to factor in the possibility of process changes, i.e., concept drifts, that could have taken place. In this section, we present the results of analysis of concept drifts in event logs pertaining to one of the processes related to permits for advertisements. If a person/organization wants to advertise on a building in the Netherlands, for example, on a billboard or an illuminated sign, a permit is needed usually, which can be obtained from the local municipality.

We considered an event log containing 116 cases and 2335 events referring to 25 activities. The cases pertain to permit requests for placing advertisements spanning over the period between July 7, 2003 and March 18, 2008. We considered the J measure feature on the follows relation for all activity pairs using a window of size 10. This choice of window size was made based on the characteristics of the process. The process has four high-level subprocesses: 1) application and initial checks; 2) regulation compliance checks; 3) decision and administration; and 4) enforcement, with clear dependencies between them. One subprocess cannot start until the previous one finished. Therefore, the dependencies between activities are primarily manifested between one subprocess and the initial few activities of its immediate successor. The event log contains 25 event classes (distinct activities) with each subprocess on an average defined over six activities. Because the dependencies are mostly reflected in one subprocess and the initial few activities of the next subprocess, a window size of 10 is deemed appropriate. In fact, we have tried using other window sizes larger than 10 as well; however, we did not notice any difference in performance with respect to change detection and change localization. Because a smaller window size is computationally efficient, we report the results on window size of 10.

The J measure values of each activity pair define a vector of size 116, corresponding to the traces in the event log. The univariate KS test is applied on each of these vectors using a population size of 10. Fig. 19 shows the average significance probability of the KS test on all activity pairs. We observe four troughs formed at indexes 42, 74, 84, and 103. These troughs signify a change in behavior in the traces preceding and succeeding them. Among the four troughs, the one at index 42 is particularly significant. Fig. 19 also shows the start timestamps (October 4, 2004, October 27, 2005, February 13, 2006, and August 31, 2006, respectively) of the cases corresponding to these troughs.

With the four change points, we split the log into five partitions, the first, $\mathcal{L}_1$, containing the traces from the beginning until the first change point (i.e., traces 1–42), the second, $\mathcal{L}_2$, containing the traces between the first and second change points (i.e., traces 43–74) and so on. Fig. 20 shows the process model discovered using the Heuristic miner [56] on the event log $\mathcal{L}_1$. The process can be divided into four high-level subprocedures, as shown in the figure, and are listed as

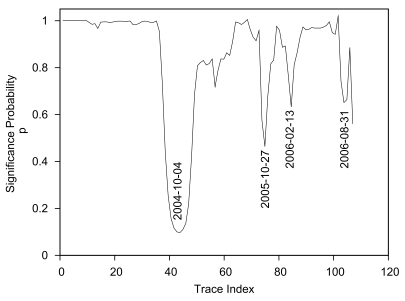


Fig. 19. Average significance probability (over all activity pairs) of KS test on J measure. The population size for the KS test is 10. There are four troughs signifying a change in behavior.

follows.

1) Upon submission of an application, the municipality acknowledges the receipt of documents and (optionally) tests for its completeness.
2) Then, the municipality proceeds with a follow-up procedure that verifies whether the application and submitted documents are in compliance with the regulations.
3) With the investigations, then the municipality makes a decision on the application and informs the applicant with the decision along with a fee letter.
4) Finally, the municipality registers the advertisements placed and enforces them.

Fig. 20(b) shows the process model discovered using the Heuristic miner [56] on the event log $\mathcal{L}_2$. The figure highlights regions that differ from the process model in Fig. 20(a). There are two changes in this model with respect to the previous one. The first change is related to the checking for completeness of the registered documents. In the initial process model [Fig. 20(a)], this check was not mandatory (only two of the 43 applications were checked for completeness). The municipality changed this process by making the checks mandatory before proceeding. The second change is the introduction of a new activity 'End procedure; enforcement is next,' as shown in Fig. 20(b). The initial process model had only the activity 'End procedure, possibly choose enforcement' where as the new model has both these activities. Similar changes have been observed in the rest of the models. We do not provide them here because of space constraints. For a more detailed discussion on this case study refer [57].

The experiments demonstrate that the features and the framework proposed in this paper for handling concept drifts show significant promise in detecting behavioral changes by analyzing event logs.

C. Time Complexity

In this section, we assess the time complexity of the proposed approach. The feature extraction is dependent on the size of the event log (i.e., the number of events). The feature values for all of the features proposed in this paper can be extracted in linear time with respect to the size of the event log. Fig. 21(a) shows the average time along with 95% confidence intervals (over five independent runs) for extracting the J measure

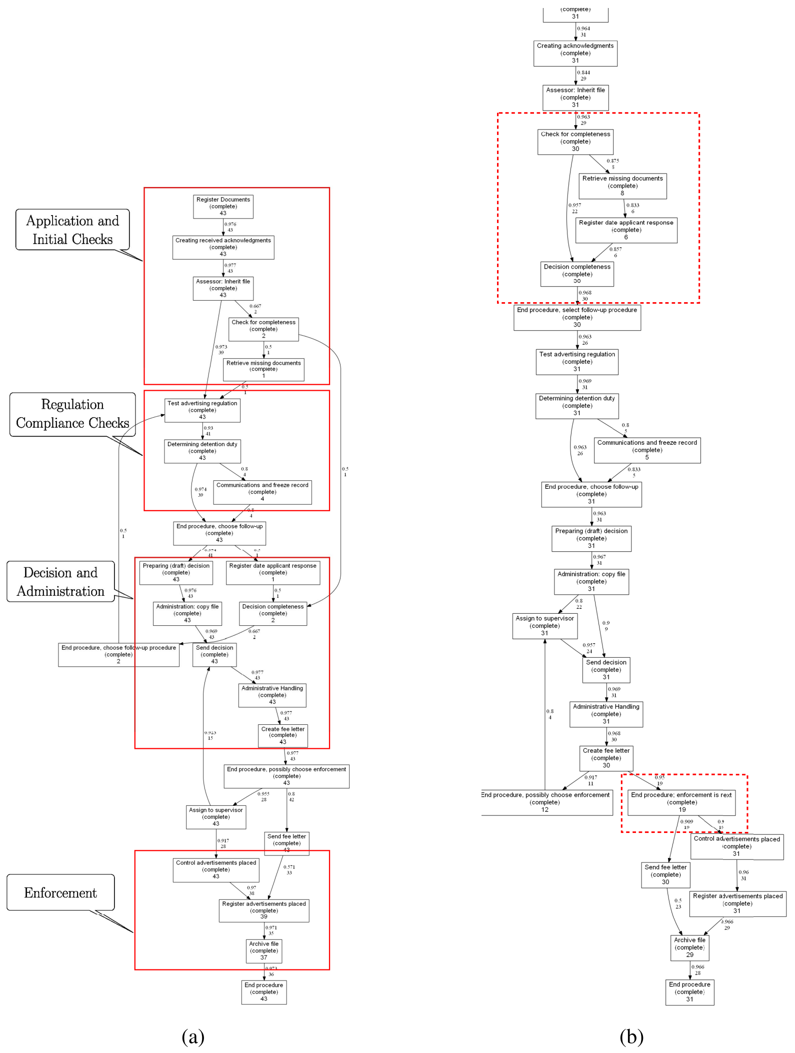

Application and Initial Checks
Regulation Compliance Checks
Decision and Administration
Enforcement
(a)
(b)

Fig. 20. Heuristic net of the permit process for advertisements discovered using the event log $\mathcal{L}_1$ (a) marked regions: high-level subprocedures in this process and (b) dashed rectangles: regions of change with respect to the previous model in (a). For a more detailed discussion on this case study refer to [57].

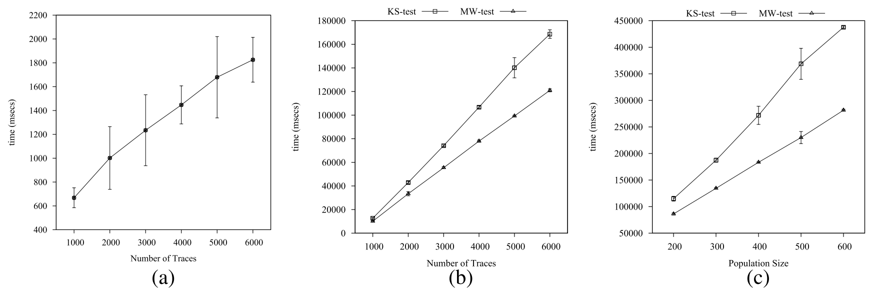


Fig. 21. Time complexity analysis: average time along with 95% confidence intervals (over five independent runs) for feature extraction and hypothesis tests. (a) J-measure. (b) Influence of number of traces. (c) Influence of population size.

feature for varying sizes of event log. For this experiment, we considered the first 1000, 2000, 3000, 4000, 5000, and 6000 traces in the juxtaposed event log (of the insurance claim example used in Section X-A). Because the average number of events is the same in each of these logs, we depict the number of traces in the x-axis. We can observe that time complexity varies linearly with respect to the size of the log. The hypothesis tests on the other hand depend on the population size and the number of traces in the event log (because the data stream of feature values is dependent on the number of traces). The number of hypothesis tests to be performed for a given data stream of n values (i.e., n traces) using a population size of p is $n - 2 * p + 1$. The time for each hypothesis test is dependent on the specific hypothesis test adopted and the population size. Fig. 21(b) shows the average time along with the 95% confidence intervals (over five independent runs) for the KS and MW tests using a population size of 300 over all activity pairs on the J measure feature for varying number of traces. Fig. 21(c) shows the average time along with the 95% confidence intervals (over five independent runs) for the KS and MW tests over all activity pairs on the J measure feature for varying population sizes (we have used the event log with 6000 traces for this experiment). We can observe that the time complexity of the hypothesis tests varies linearly with respect to both the number of traces and the population size. Because our approach is primarily intended for offline use, the overall time for change detection is reasonable. Furthermore, we have considered all activity pairs as features and the average significance p-values over them for this experiment. The overall computation time can be further reduced by robust feature selection techniques.

XI. Conclusion

In this paper, we have introduced the topic of concept drift in process mining, i.e., analyzing process changes based on event logs. We proposed feature sets and techniques to effectively detect the changes in event logs and identify the regions of change in a process. Our initial results show that heterogeneity of cases arising because of process changes can be effectively dealt with by detecting concept drifts. Once change points are identified, the event log can be partitioned and analyzed. This is the first step in the direction of dealing with changes in any process monitoring and analysis efforts. We have considered changes only with respect to the control-flow perspective manifested as sudden and gradual drifts. Therefore, our analysis should only be observed as the starting point for a new subfield in the process mining domain and there are lots of challenges that still need to be addressed. Some of these challenges include.

1) *Change-pattern specific features:* In this paper, we presented very generic features (based on follows/precedes relation). These features are neither complete nor sufficient to detect all classes of changes. An important direction of research would be to define features catering to different classes of changes and investigate their effectiveness. A taxonomy/classification of change patterns and the appropriate features for detecting changes with respect to those patterns are needed.
2) *Feature selection:* The feature sets presented in this paper result in a large number of features. For example, the activity relation count feature type generates $3 \times |\mathcal{A}|$ features whereas the WC and J measure generate $|\mathcal{A}|^2$ features (corresponding to all activity pairs). On the one hand, such high dimensionality makes analysis intractable for most real-life logs. On the other hand, changes being typically concentrated in a small region of a process make it unnecessary to consider all features. There is a need for tailored dimensionality reduction techniques [44], [45] that can efficiently select the most appropriate features.
3) *Holistic approaches:* In this paper, we discussed ideas on change detection and localization in the context of sudden and gradual changes to the control-flow perspective of a process. As mentioned in Section IV, the data and resource perspectives are also, however, equally important. Features and techniques that can enable the detection of changes in these other perspectives need to be discovered. Furthermore, there could be instances where more than one perspective (e.g., both control and resource) change simultaneously. Hybrid approaches considering all aspects of change holistically need to be developed.
4) *Recurring drifts:* When dealing with recurring drifts, in addition to change point detection and change localization, it is important to identify the variant(s) that recur. This requires robust metrics to assess the similarity between process variants and/or event logs.
5) *Change process discovery:* As mentioned earlier, after detecting the change points and the regions of change, it is necessary to put them together in perspective. Organizations would be interested in discovering the evolution of change (e.g., as an animation depicting how the process has changed/evolved over time). In addition, there are other applications such as deriving a configurable model for the process variants. A configurable process model describes a family of similar process models [58]. The process variants discovered using concept drift can be merged to derive a configurable process model.
6) *Sample complexity:* Sample complexity refers to the number of traces (size of the event log) needed to detect, localize, and characterize changes within acceptable error bounds. This should be sensitive to the variability in processes (in the manifestation of various process model constructs used), nature of changes, their influence and manifestation in traces, and the feature space and algorithms used for detecting drifts. On a broader note, the topic of sample complexity is relevant to all facets of process mining and is hardly addressed. For example, it would be interesting to know the lower bound on the number of traces required to discover a process model with a desired fitness.
7) *Online (on-the-fly) drift detection:* In this paper, we have looked at detecting drifts in an offline setting, i.e., for postmortem analysis. Although detecting concept drifts is important for offline analysis, it is more interesting and appropriate for online analysis. We believe the proposed framework to be applicable even for online analysis. Few new challenges, however, emerge, e.g., the number of samples required remains an issue. In addition, we need additional computational power and efficient techniques to do such analysis in near real time.

References

[1] (2010). *All-in-one Permit for Physical Aspects: (Omgevingsvergunning) in a Nutshell* [Online]. Available: http://www.answersforbusiness.nl/regulation/all-in-one-permit-physical-aspects

[2] United States Code. (2002, Jul.). *Sarbanes-Oxley Act of 2002, PL 107-204, 116 Stat 745* [Online]. Available: http://files.findlaw.com/news.findlaw.com/cnn/docs/gwbush/sarbanesoxley072302.pdf

[3] W. M. P. van der Aalst, M. Rosemann, and M. Dumas, “Deadline-based escalation in process-aware information systems,” *Decision Support Syst.*, vol. 43, no. 2, pp. 492–511, 2011.

[4] M. Dumas, W. M. P. van der Aalst, and A. H. M. Ter Hofstede, *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. New York, NY, USA: Wiley, 2005.

[5] W. M. P. van der Aalst and K. M. van Hee, *Workflow Management: Models, Methods, and Systems*. Cambridge, MA, USA: MIT Press, 2004.

[6] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. New York, NY, USA: Springer-Verlag, 2011.

[7] B. F. van Dongen and W. M. P. van der Aalst, "A meta model for process mining data," in *Proc. CAiSE Workshops (EMOI-INTEROP Workshop)*, vol. 2. 2005, pp. 309–320.

[8] C. W. Günther, (2009). *XES Standard Definition* [Onlilne]. Available: http://www.xes-standard.org

[9] F. Daniel, S. Dustdar, and K. Barkaoui, "Process mining manifesto," in *BPM 2011 Workshops*, vol. 99. New York, NY, USA: Springer-Verlag, 2011, pp. 169–194.

[10] R. P. J. C. Bose, W. M. P. van der Aalst, I. Žliobaitė, and M. Pechenizkiy, "Handling concept drift in process mining," in *Proc. Int. CAiSE*, 2011, pp. 391–405.

[11] J. Carmona and R. Gavaldà, "Online techniques for dealing with concept drift in process mining," in *Proc. Int. Conf. IDA*, 2012, pp. 90–102.

[12] J. Schlimmer and R. Granger, "Beyond incremental processing: Tracking concept drift," in *Proc. 15th Nat. Conf. Artif. Intell.*, vol. 1. 1986, pp. 502–507.

[13] A. Bifet and R. Kirkby. (2011). *Data Stream Mining: A Practical Approach*, University of Waikato, Waikato, New Zealand [Online]. Available: http://www.cs.waikato.ac.nz/~abifet/MOA/StreamMining.pdf

[14] I. Žliobaitė, "Learning under concept drift: An Overview," *CoRR*, vol. abs/1010.4784, 2010 [Online]. Available: http://arxiv.org/abs/1010.4784

[15] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. SBIA*, 2004, pp. 286–295.

[16] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proc. 7th SIAM Int. Conf. Data Mining (SDM)*, 2007, pp. 443–448.

[17] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognit. Lett.*, vol. 33, no. 2, pp. 191–198, 2012.

[18] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *Proc. 10th Int. Conf. Discovery Sci.*. 2007, pp. 264–269.

[19] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers–Part I: Detecting nonstationary changes," *IEEE Trans. Neural Netw.*, vol. 19, no. 7, pp. 1145–1153, Jul. 2008.

[20] C. Alippi, G. Boracchi, and M. Roveri, "Just-in-time classifiers for recurrent concepts," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 4, pp. 620–634, Apr. 2013.

[21] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, Jan. 2007.

[22] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, Oct. 2011.

[23] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996.

[24] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," *Knowl. Based Syst.*, vol. 18, nos. 4–5, pp. 187–195, Aug. 2005.

[25] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Handling local concept drift with dynamic integration of classifiers: Domain of antibiotic resistance in nosocomial infections," in *Proc. 19th IEEE Int. Symp. CBMS*, Nov. 2006, pp. 679–684.

[26] M. Pechenizkiy, J. Bakker, I. Žliobaitė, A. Ivannikov, and T. Kärkkäinen, "Online mass flow prediction in CFB boilers with explicit detection of sudden concept drift," *SIGKDD Explorations*, vol. 11, no. 2, pp. 109–116, 2009.

[27] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, nos. 1–2, pp. 100–115, 1954.

[28] N. Mulyar, "Patterns for process-aware information systems: An approach based on colored Petri nets," Ph.D. dissertation, Dept. Comput. Sci., Univ. Technol., Eindhoven, The Netherlands, 2009.

[29] B. Weber, S. Rinderle, and M. Reichert, "Change patterns and change support features in process-aware information systems," in *Proc. 19th Int.*, 2007, pp. 574–588.

[30] G. Regev, P. Soffer, and R. Schmidt, "Taxonomy of flexibility in business processes," in *Proc. 7th Workshop BPMDS*, 2006, pp. 1–4.

[31] H. Schonenberg, R. Mans, N. Russell, N. Mulyar, and W. M. P. van der Aalst, "Process flexibility: A survey of contemporary approaches," in *Proc. Adv. Enterprise Eng. I*, 2008, pp. 16–30.

[32] K. Ploesser, J. C. Recker, and M. Rosemann, "Towards a classification and lifecycle of business process change," in *Proc. BPMDS*, vol. 8. 2008, pp. 1–9.

[33] C. W. Günther, S. Rinderle-Ma, M. Reichert, and W. M. P. van der Aalst, "Using process mining to learn from process changes in evolutionary systems," *Int. J. Business Process Integr. Manag.*, vol. 3, no. 1, pp. 61–78, 2008.

[34] M. van Leeuwen and A. Siebes, "StreamKrimp: Detecting change in data streams," in *Proc. Mach. Learn. Knowl. Discovery Databases*, 2008, pp. 672–687.

[35] I. Žliobaitė and M. Pechenizkiy. (2010). *Handling Concept Drift in Information Systems* [Online]. Available: http://sites.google.com/site/zliobaite/CD_applications_2010.pdf

[36] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. 2003, pp. 226–235.

[37] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, Apr. 2013, doi: 10.1109/TNNLS.2013.2251352.

[38] W. M. P. van der Aalst, A. Adriansyah, and B. Dongen, "Replaying history on process models for conformance checking and performance analysis," *WIREs Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 182–192, 2012.

[39] M. Hammer, *Beyond Reengineering: How the Process-Centered Organization is Changing Our Work and Our Lives*. New York, NY, USA: Harper business, 1996.

[40] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 5, pp. 730–742, May 2010.

[41] P. Smyth and R. M. Goodman, *Rule Induction Using Information Theory*. Washington, DC, USA: AAAS Press, 1991, pp. 159–176.

[42] N. M. Blachman, "The amount of information that y gives about X," *IEEE Trans. Inf. Theory*, vol. 14, no. 1, pp. 27–31, Jan. 1968.

[43] D. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. London, U.K.: Chapman & Hall/CRC, 2004.

[44] I. K. Fodor, "A survey of dimensionality reduction techniques," in *Proc. Center Appl. Sci. Comput., Lawrence Livermore Nat. Lab.*, 2002, pp. 1–24.

[45] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[46] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed., New York, NY, USA: Springer-Verlag, 2002.

[47] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *Proc. 7th ACM SIGKDD Int. Conf. Mining*, 2001, pp. 245–250.

[48] W. M. P. van der Aalst and A. H. M. ter Hofstede, "YAWL: Yet another workflow language," *Inf. Syst.*, vol. 30, no. 4, pp. 245–275, 2005.

[49] W. M. P. van der Aalst, "Re-engineering knock-out processes," *Decision Support Syst.*, vol. 30, no. 4, pp. 451–468, 2001.

[50] K. Jensen and L. M. Kristensen, *Colored Petri Nets: Modeling and Validation of Concurrent Systems*. New York, NY, USA: Springer-Verlag, 2009.

[51] CoSeLog, (2013). *Configurable Services for Local Governments*, Germany [Online]. Available: http://www.win.tue.nl/coselog

[52] W. M. P. van der Aalst, "Configurable services in the cloud: Supporting variability while enabling cross-organizational process mining," in *On the Move to Meaningful Internet Systems (OTM 2010)*, LNCS 6426. New York, NY, USA: Springer-Verlag, Jan. 2010, pp. 8–25.

[53] W. M. P. van der Aalst, "Intra- and inter-organizational process mining: Discovering processes within and between organizations," in *Proc. Pract. Enterprise Model.*, 2011, pp. 1–11.

[54] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "Towards cross-organizational process mining in collections of process models and their executions," in *Proc. Int. Workshop PMC*, 2011, pp. 1–14.

[55] J. J. C. L. Vogelaar, H. M. W. Verbeek, and W. M. P. van der Aalst, "Comparing business processes to determine the feasibility of configurable models: A case study," in *Proc. Int. Workshop PMC*, 2011, pp. 1–12.

[56] A. J. M. M. Weijters and W. M. P. van der Aalst, "Rediscovering workflow models from event-based data using little thumb," *Integr. Comput. Aided Eng.*, vol. 10, no. 2, pp. 151–162, 2003.

[57] R. P. J. C. Bose, W. M. P. van der Aalst, I. Žliobaitė, and M. Pechenizkiy, "Dealing with concept drifts in process mining: A case study in a Dutch municipality," BPM Center, Univ. Technol., Singapore, Tech. Rep. BPM-13-13, 2013.

[58] W. M. P. van der Aalst, N. Lohmann, and M. L. Rosa, "Ensuring correctness during process configuration via partner synthesis," *Inf. Syst.*, vol. 37, no. 6, pp. 574–592, 2012.

R. P. Jagadeesh Chandra Bose received the Ph.D. (*cum laude*) degree in process mining from Technische Universiteit Eindhoven, Eindhoven, The Netherlands.

He is a Research Scientist with Xerox Research Center India, Bangalore, India. He has over eight years of research experience in the industry, and he has executed projects in the areas of software engineering, health care, business analytics, and knowledge management. He has co-authored over 30 publications and five patent filings. His current research interests include process mining, business process management, business intelligence (analytics), machine learning, data mining, and bioinformatics.

Dr. Bose has received various awards for excellence in academics.

Wil M. P. van der Aalst received the Doctor Honoris Causa degree from Hasselt University, Diepenbeek, Belgium, in 2012.

He is a Full Professor of information systems with the Technische Universiteit Eindhoven (TU/e), Eindhoven, The Netherlands. He is the Academic Supervisor of the International Laboratory of Process-Aware Information Systems, National Research University, Higher School of Economics, Moscow, Russia. Since 2003, he has held a part-time appointment with the Queensland University of Technology, Queensland, Australia. His ideas have influenced researchers, software developers, and standardization committees working on process support. In 2013, he was a Distinguished University Professor at TU/e. His papers are highly cited (he has an H-index of more than 103 according to Google Scholar, making him the European computer scientist with the highest H-index). He has published more than 160 journal papers, 17 books (as author or editor), 300 refereed conference/workshop publications, and 50 book chapters. His current research interests include workflow management, process mining, Petri nets, business process management, process modeling, and process analysis.

Dr. van der Aalst is a member of the Royal Holland Society of Sciences and Humanities (Koninklijke Hollandsche Maatschappij der Wetenschappen) and the Academy of Europe (Academia Europaea).

Indrė Žliobaitė is a Research Scientist with Aalto University, Aalto, Finland. Her current research interests include predictive modeling from evolving streaming data, change detection, and predictive analytics applications. For further information see http://zliobaite.googlepages.com.

Mykola Pechenizkiy received the Ph.D. degree in computer science and information systems from the University of Jyvaskyla, Jyvaskyla, Finland, in 2005.

He is an Assistant Professor with the Department of Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands. He has co-authored over 70 peer-reviewed publications and has been organizing several workshops (HaCDAIS at ECML/PKDD in 2010, LEMEDS at AIME in 2011), conferences (IEEE CBMS in 2012, EDM in 2011, IEEE CBMS in 2008, and BNAIC in 2009) and tutorials (at ECML/PKDD in 2012, PAKDD in 2011, IEEE CBMS in 2010, and ECML/PKDD in 2010). He has co-edited the *Handbook of Educational Data Mining*. His current research interests include data mining and data-driven intelligence, and its application to various (adaptive) information systems serving industry, commerce, medicine, and education.

Dr. Pechenizkiy has served as a Guest Editor of the special issues in *SIGKDD Explorations*, Elsevier's *DKE and AIIM*, and Springer's *Evolving Systems Journals*. Currently, he takes a leading role in NWO HaCDAIS, STW CAPA, EIT ICT Labs Stress at Work, and NL Agency CoDaK.