Handling Concept Drift in Process Mining

R.P. Jagadeesh Chandra Bose[1,2], Wil M.P. van der Aalst[1], Indrė Žliobaitė[1], and Mykola Pechenizkiy[1]

[1] Department of Mathematics and Computer Science, University of Technology, Eindhoven, The Netherlands
[2] Philips Healthcare, Veenpluis 5–6, Best, The Netherlands
{j.c.b.rantham.prabhakara,w.m.p.v.d.aalst,m.pechenizkiy}@tue.nl, zliobaite@gmail.com

Abstract. Operational processes need to change to adapt to changing circumstances, e.g., new legislation, extreme variations in supply and demand, seasonal effects, etc. While the topic of flexibility is well-researched in the BPM domain, contemporary *process mining* approaches assume the process to be in steady state. 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. Obviously, this is often not the case due to the phenomenon known as *concept drift*. While cases are being handled, the process itself may be changing. This paper presents an approach to analyze such *second-order dynamics*. The approach has been implemented in ProM[1] and evaluated by analyzing an evolving process.

Keywords: process mining, concept drift, flexibility, change patterns.

1 Introduction

In order to retain their competitive advantage in today's dynamic marketplace, it is increasingly necessary for enterprises to streamline their processes so as to reduce costs and to improve performance. Moreover, today's customers expect organizations to be flexible and adapt to changing circumstances. New legislation is also forcing organizations to change their processes. It is clear that the economic success of an organization is highly dependent on its ability to react 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 have been extended to be able to flexibly adapt to changes in the process. State-of-the-art Workflow Management (WFM) and BPM systems provide flexibility. Moreover, in processes not driven by WFM/BPM systems there is even more flexibility as processes are controlled by people.

Although flexibility and change have been studied in-depth in the context of WFM and BPM systems, *existing process mining techniques assume processes*

[1] ProM is an extensible framework that provides a comprehensive set of tools/plugins for the discovery and analysis of process models from event logs. See `http://www.processmining.org` for more information and to download ProM.

to be in steady state. Starting point for process mining is an event log containing a sequence of business events recorded by one or more information systems. Based on such an event log, processes can be discovered. Today's process discovery techniques are able to extract meaningful process models from event logs not containing any explicit process information. Using ProM, 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 steady state: while analyzing the process, changes can take place. For example, governmental and insurance organizations reduce the fraction of cases being checked when there is too much work in the pipeline. In case of a disaster, hospitals and banks change their operating procedures etc. Such changes are indirectly reflected in the event log. Moreover, analyzing such changes is of the utmost importance when supporting or improving operational processes.

In the data mining and machine learning communities, such second-order dynamics are referred to as *concept drift*, and has been studied in both supervised and unsupervised settings. Concept drift has been shown to be important in many applications and several successful stories have been reported in the literature [1,2,3]. However, existing work tends to focus on simple structures such as changing variables rather than changes to complex artifacts such as process models describing concurrency, choices, loops, cancelation, etc. In handling concept drifts in process mining, the following three main problems can be identified:

1. *Change (Point) Detection:* The first and most fundamental problem is to detect concept drift in processes, i.e., 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.
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 (for example, control-flow, data, resource, sudden, gradual etc.) and the exact change in itself.
3. *Unravel Process Evolution:* 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).

In this paper, we focus on the first two problems. We propose features and techniques to detect changes (drifts), change points, and change localization in event logs from a control-flow perspective. The techniques proposed in this paper show significant promise in handling concept drifts. We further provide an outlook on some of the topics in concept drift and believe that this niche area, with its broad scope and relevance, evokes lots of interest in the research community.

The remainder of this paper is structured as follows. Related work is presented in Section 2. Section 3 describes the various aspects and nature of change. Section 4 introduces various features and techniques for detecting drifts in event logs. Section 5 describes the effectiveness of the features and techniques proposed in this paper in discovering change points and localization of changes through a case study. In Section 6, we project an outlook on some of the open research questions and directions in this area. The paper ends with some conclusions in Section 7.

2 Related Work

Over the last two decades many researchers have been working on process flexibility, e.g., making workflow systems adaptive. In [4,5] collections of typical change patterns are described. In [6,7] extensive taxonomies of the various flexibility approaches and mechanisms are provided. Ploesser et al. [8] have classified business process changes into three broad categories viz., sudden, anticipatory and evolutionary. This classification is used in this paper, but now in the context of event logs.

Despite that many publications on flexibility, most process mining techniques assume a steady state process. A notable exception is the approach by Günther et al. [9]. This approach uses process mining to provide an aggregated overview of all changes happened so far. However, this approach assumes that change logs are available, i.e., modifications of the workflow model are recorded. At this point in time very few information systems provide change logs. Therefore, this paper focuses on concept drift in process mining assuming only an event log as input. Concept drift refers to changes in the target variable(s)/concept induced by contextual shifts over time [10]. While the topic is well-studied in various branches of the data mining and machine learning community, the problem of concept drift has not been studied in the process mining community. While experiences from data mining and machine learning can be used to investigate concept drift in process mining, existing techniques cannot be used due to the complexity of process models and the nature of process change.

3 Aspects and Nature of Change in Business Processes

Three important perspectives in the context of business processes are the control-flow, data and resource perspective. One or more of these perspectives may be subjected to a change.

– *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 software engineering, there exist *change patterns* capturing the common control-flow changes [4]. 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 the fee after the processing and acceptance of an application can now change their process to enforce the payment of fee before the processing of an application. Here the *reordering* change pattern had been applied on the payment and 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 PAIS, various control-flow change patterns have been proposed in [4,5]. 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 could have been changed. 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 due to the organization’s decision to increase the claim limit. The structure of the process remains intact but the routing of cases changes.
– *Data Perspective:* This class of changes refer to the changes in the requirement, usage, and generation of data in a process. Tasks may produce or require information/data. An example of change in a data perspective is enabling the execution of a task without the requirement of an otherwise needed data element d.
– *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 in what roles in a process. As another example, certain execution paths in a process could be enabled (disabled) upon the availability (non-availability) of resources. Furthermore, resources tend to work in a particular manner and this bias may change over time. For example, a resource can have a bias of executing a set of parallel activities in a specific sequential order. Such biases could be more prominent when a limited number of resources are available; the addition of new resources can remove this bias.

Based on the duration for which a change is active, one can classify changes into momentary and permanent. Momentary changes are short-lived and affect only a very few cases while permanent changes are persistent and stay for a while [6]. In this paper, we consider only permanent changes. Changes are perceived to induce a drift in the concept (process behavior). We identify four classes of drifts as depicted in Fig. 1 based on how they manifest.

– *Sudden Drift:* This corresponds to a substitution of an existing process M_1 with a new process M_2 as depicted in Fig. 1(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 are typically seen in scenarios such as emergency response planning. As an example, airlines and airports changing their security processes due to a new regulation.
– *Recurring Drift:* This corresponds to the scenario where a set of processes reappear after some time (substituted back and forth) as depicted in Fig. 1(b). It is quite natural to see 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 non-periodic. An example of a non-periodic recurrence is a deployment of a process subject to market conditions. The point of deployment and duration of deployment are both dependent on external factors (here, the market conditions).

– *Gradual Drift:* This refers to the scenario as depicted in Fig. 1(c) 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. However, this process is applicable only for orders taken henceforth. All previous orders still have to follow the older delivery process.
– *Incremental Drift:* This refers to the scenario where a substitution of process M_1 with M_N is done via smaller incremental changes as depicted in Fig. 1(d). This class of drifts is more pronounced in organizations adopting agile business process management methodology.

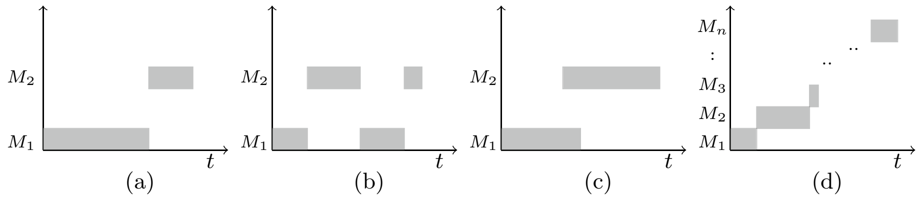


Fig. 1. Different types of drifts. (a) sudden drift (b) recurring drift (c) gradual drift and (d) incremental drift. X-axis indicate time and Y-axis indicate process variants. Shaded rectangles depict process instances.

4 Approaches to Detecting Drifts in Event Logs

We propose approaches to detect potential control-flow changes in a process manifested as sudden drifts over a period of time by analyzing its event log. Detecting drifts in data and resource perspectives and in the contexts of gradual, recurring and incremental drifts is beyond the scope of this paper.

4.1 Causal Footprints

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. For any pair of activities, `a` and `b` $\in \Sigma$, one can determine whether they exhibit either *always*, *never*, or *sometimes* follows/precedes relationship. 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 or in none of the traces, then we say that `b` *sometimes follows* `a`, and `b` *never follows* `a` respectively. Consider an event log $\mathcal{L} = \{\texttt{acaebfh}, \texttt{ahijebd}, \texttt{aeghijk}\}$ containing three traces defined over Σ ={`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`. The variants of *precedes* relation can be defined on similar lines. The *follows/precedes* relationship is rich enough to reveal many control flow changes in a process. In the next section, we exploit this relationship and define various features for change detection.

4.2 Features Capturing the Manifestation of Activity Relationships

We distinguish between two classes of features (i) global features and (ii) local features. Global features are defined over an event log while local features can be defined at a trace level. Based on the follows (precedes) relation, we propose two global features viz., Relation Type Count and Relation Entropy, and two local features viz., Window Count and J-measure. These features are defined as follows:

– *Relation Type Count (RC):* The relation type count with respect to follows (precedes) relation is a function $f_{\mathrm{RC}} : \Sigma \rightarrow \mathbb{N}_0^3$ defined over the set of activities. f_{RC} of an activity, $\mathtt{b} \in \Sigma$ with respect to follows (precedes) relation over an event log $\mathcal{L}$ is a triple $\langle c_a, c_s, c_n \rangle$ where c_a, c_s, and c_n are the number of activities in Σ that always, sometimes, and never follow (precede) b in $\mathcal{L}$ respectively. For the event log $\mathcal{L}$ mentioned above, $f_{\mathrm{RC}}(\mathtt{a}) = \langle 2, 9, 0 \rangle$ since e and h always follows a while all other activities in $\Sigma \setminus \{\mathtt{e}, \mathtt{h}\}$ sometimes follows a. $f_{\mathrm{RC}}(\mathtt{i}) = \langle 1, 4, 6 \rangle$ since only j always follows i; b, d, e, and k sometimes follows i while a, c, f, g, h and i never follows i.
For an event log containing $|\Sigma|$ activities, this results in a feature vector of dimension $3|\Sigma|$ (if either follows or precedes relation is considered) or $2 \times 3|\Sigma|$ (if both follows and precedes relation are considered).
– *Relation Entropy (RE):* The relation entropy with respect to follows (precedes) relation is a function $f_{\mathrm{RE}} : \Sigma \rightarrow \mathbb{R}^+$ defined over the set of activities. f_{RE} of an activity, $\mathtt{b} \in \Sigma$ with respect to follows (precedes) relation is the entropy of the relation type count metric. In other words, $f_{\mathrm{RE}}(\mathtt{b}) = -p_a \log p_a - p_s \log p_s - p_n \log p_n$ where $p_a = c_a/|\Sigma|, p_s = c_s/|\Sigma|$, and $p_n = c_n/|\Sigma|$.
For the above example event log $\mathcal{L}$, $f_{\mathrm{RE}}(\mathtt{a}) = 0.68$ (corresponding to $f_{\mathrm{RC}}(\mathtt{a}) = \langle 2, 9, 0 \rangle$) and $f_{\mathrm{RE}}(\mathtt{i}) = 1.32$ (corresponding to $f_{\mathrm{RC}}(\mathtt{i}) = \langle 1, 4, 6 \rangle$). For an event log containing $|\Sigma|$ activities, this results in a feature vector of dimension $|\Sigma|$ or $2 \times |\Sigma|$ depending on whether either or both of follows/precedes relation is considered.
– *Window Count (WC):* The window count with respect to follows (precedes) relation is a function $f_{\mathrm{WC}} : \Sigma \times \Sigma \rightarrow \mathbb{N}_0$ defined over the set of activity pairs. Given a trace $\mathbf{t}$ and a window of size l, let S_l be the set of all subsequences $\mathbf{t}(i, i+l-1)$, such that $\mathbf{t}(i) = \mathtt{a}$ and there exists a j such that $i < j < i + l$ and $\mathbf{t}(j) = \mathtt{b}$. The window count of the relation b *follows* a is defined as the number of sequences of length l in which b follows a. In other words, $f_{\mathrm{WC}}(\mathtt{a}, \mathtt{b}) = |S_l|$.
For the above example event log $\mathcal{L}$, using a window of size $l = 4$, $f_{\mathrm{WC}}(\mathtt{a}, \mathtt{b}) = 1$ for trace acaebfh and 0 for traces ahijebd and aeghijk.
– *J-Measure:* Smyth and Goodman [11] have proposed a metric called J-measure based on [12] 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 one 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 $f_{\mathrm{J}} : \Sigma \times \Sigma \rightarrow \mathbb{R}^+$ defined over the set of activity pairs. Let $p(\mathtt{a})$ and $p(\mathtt{b})$ denote the probability of occurrence of activities a and b respectively in a trace $\mathbf{t}$. Let $p_l(\mathtt{a}F\mathtt{b})$ denote the probability that b follows

a within a window of size l. Then the J-measure is defined as $f_J(\mathtt{a},\mathtt{b}) = p(\mathtt{a})\mathrm{CE}_l(\mathtt{a}F\mathtt{b})$ where $\mathrm{CE}_l(\mathtt{a}F\mathtt{b})$ denotes the cross-entropy of a and b (b follows a within a window of size l) and is defined as

$$\mathrm{CE}_l(\mathtt{a}F\mathtt{b}) = p_l(\mathtt{a}F\mathtt{b})\log\left(\frac{p_l(\mathtt{a}F\mathtt{b})}{p(\mathtt{b})}\right) + (1-p_l(\mathtt{a}F\mathtt{b}))\log\left(\frac{1-p_l(\mathtt{a}F\mathtt{b})}{1-p(\mathtt{b})}\right)$$

The J-measure of b follows a for trace acaebfh using a window of size $l = 4$ is $f_J(\mathtt{a},\mathtt{b}) = 0.147$.

Though local features are defined at a trace level, it is easy to lift them to the level of an entire event log.

4.3 Statistical Hypothesis Tests to Detect Drifts

One can consider an event log $\mathcal{L}$ as a time series of traces (traces ordered on their arrival time). Fig. 2 depicts such a perspective on an event log along with change points. An event log can be split into sub-logs of s traces each. We can consider either overlapping or non-overlapping windows when creating such sub-logs. Fig. 2 depicts the scenario where two subsequent sub-logs do not overlap. In this case, we have $k = \lceil \frac{n}{s} \rceil$ sub-logs for n traces. One can estimate the feature values for each trace separately (local features) or cumulatively over a subset of traces (local and global features) and generate a dataset defined by a matrix/vector of feature values over a sub-log/trace. For example, the relation count feature type will generate a dataset $\mathcal{D}$ of size $k \times 3|\Sigma|$ when either the follows/precedes relation counts of all activities are considered over $\mathcal{L}$. Instead, if the follows/precedes relation count of an individual activity is considered in isolation, it generates a dataset of size $k \times 3$ for $\mathcal{L}$. The J-measure generates a scalar value for each trace (sub-log) when an activity pair is considered thereby generating a vector of size $n \times 1$ or $k \times 1$ (depending on whether it is measured over traces or sub-logs) over $\mathcal{L}$. If all activity pairs are considered, then a dataset of size $n \times |\Sigma|^2$ or $k \times |\Sigma|^2$ is generated.

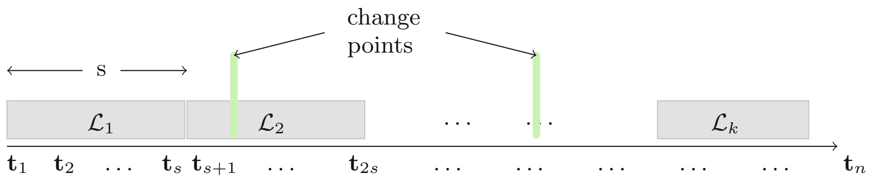

change points
s
$\mathcal{L}_1$
$\mathcal{L}_2$
...
...
$\mathcal{L}_k$
$\mathbf{t}_1$ $\mathbf{t}_2$ ... $\mathbf{t}_s$ $\mathbf{t}_{s+1}$ ... $\mathbf{t}_{2s}$ ... ... $\mathbf{t}_n$

Fig. 2. An event log and change points

We believe that there should be a characteristic difference in the manifestation of feature values in the traces (sub-logs) before and after the change points with the difference being more pronounced at the boundaries. *The goal of concept drift in process mining is then to detect the change points and the nature of changes given an event log.* 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 can distinguish between two classes of hypothesis tests (i) tests on a single population (single-sample tests) and (ii) tests on two populations (two-sample 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 as *univariate* tests while those dealing with vector data elements are called as *multi-variate* tests. For our problem, *two-sample univariate and multi-variate tests are appropriate.* The dataset $\mathcal{D}$ of feature values can be considered as a time series as depicted in Fig. 3. Each $\mathbf{d}_i \in \mathcal{D}$ corresponds to a feature value for a trace (or sub-log) and can be a scalar or a vector. *The basic idea is to consider a series of successive populations of values (of size w) and investigate if there is a significant difference between the two 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. 3 depicts a scenario where two populations $P_1 = \langle \mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_w \rangle$ and $P_2 = \langle \mathbf{d}_{w+1}, \mathbf{d}_{w+2}, \ldots, \mathbf{d}_{2w} \rangle$ of size w are considered. In the next iteration, the populations correspond to $P_1 = \langle \mathbf{d}_2, \mathbf{d}_3, \ldots, \mathbf{d}_{w+1} \rangle$ and $P_2 = \langle \mathbf{d}_{w+2}, \mathbf{d}_{w+3}, \ldots, \mathbf{d}_{2w+1} \rangle$. Given a dataset of m values, the number of population pairs will be $m - 2w + 1$.

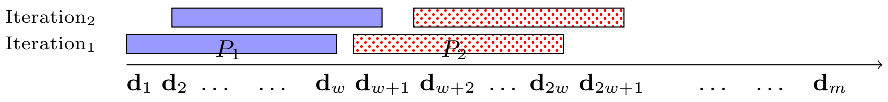

Iteration$_2$
Iteration$_1$
P_1
P_2
$\mathbf{d}_1$ $\mathbf{d}_2$ ... ... $\mathbf{d}_w$ $\mathbf{d}_{w+1}$ $\mathbf{d}_{w+2}$ ... $\mathbf{d}_{2w}$ $\mathbf{d}_{2w+1}$ ... ... $\mathbf{d}_m$

Fig. 3. Dataset of feature values considered as a time series for hypothesis tests. P_1 and P_2 are two populations of size w

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 (populations P_1 and P_2) represent two different cumulative frequency distributions?" while the *MW* test evaluates the hypothesis "Do the two independent samples have different distributions with respect to the rank-ordering of the values?". The multi-variate 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 the reader to [13] for a classic introduction to various hypothesis tests.

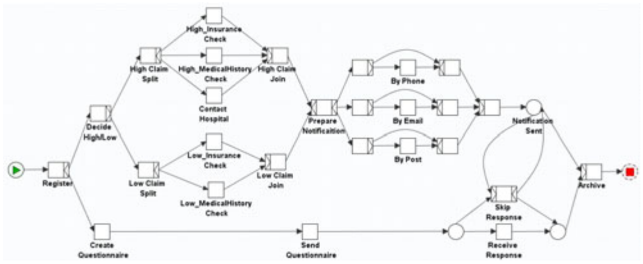
5 Case Study and Discussion

We illustrate the concepts presented in this paper with an example process. The 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 into a high or low claim. For low claims, two independent tasks, viz., check insurance and check medical history need to be executed. For high claims, three tasks need to be executed viz., check insurance, check medical history, and 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. An insurance grant 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. Three modes of notification are supported viz., by email, by telephone (fax) and by postal mail. The case should be archived upon notifying the claimant. This can be done with or without the response for the questionnaire. However, the decision of ignoring the questionnaire can only be made after a notification is sent. The case is closed upon completion of archiving task.

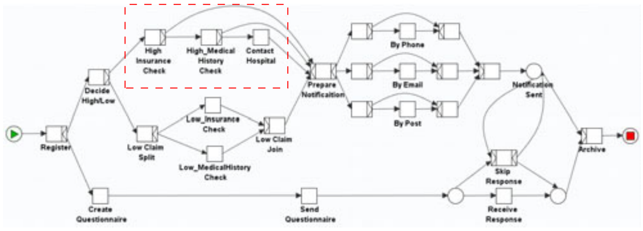
Fig. 4 depicts five variants of this process represented in YAWL [14] notation. The dashed rectangles indicate regions where a change has been done in the process model with respect to its previous variant. The changes can have various reasons. For example, in Fig. 4(a), the different checks for high insurance claims are modeled using a parallel construct. However, a claim could 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 adopted for high insurance checks as depicted in Fig. 4(b). As another example, the OR-construct pertaining to the sending of notification to claimants in Fig. 4(c) has been modified to an exclusive-or (XOR) construct in Fig. 4(d). The organization could have taken a decision to reduce their workforce as a cost-cutting measure. Due to 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.

Let us denote these process variants as M_1, M_2, M_3, M_4 and M_5. We have modeled each of these process variants in CPN tools [15] and simulated 1200 traces for each model. We created an event log $\mathcal{L}$ of 6000 traces by juxtaposing each set of the 1200 traces. The event log contains 15 activities or event classes (i.e., $|\Sigma| = 15$) and 58953 events. Given this event log $\mathcal{L}$, *our first objective is to detect the four change points pertaining to these five process variants* as depicted in Fig. 5.

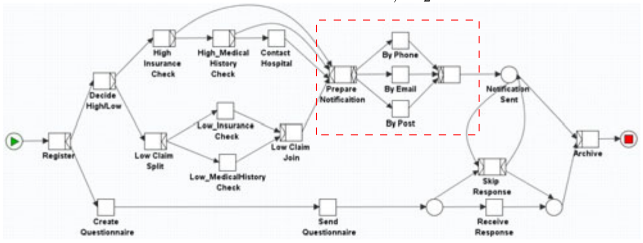
The ideas presented in this paper have been implemented as the *concept drift* plugin in ProM. We have considered global features (at sub-log level) and local features (both at trace and sub-log level) for our analysis. To facilitate this, we have split the log into 120 sub-logs using a split size of 50 traces. We have computed the relation type count (RC) of all 15 activities thereby generating a multi-variate vector of 45 features for each sub-log. We have applied the Hotelling T^2 hypothesis test on this multi-variate dataset using a moving window of size, $w = 8$. For this hypothesis test, we have randomly chosen 6 of the 45 features with a 10-fold cross validation. Fig. 6a depicts 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 values in the log. In other words, they indicate that there is drift (change) in the concept, which here corresponds to the process. It is interesting to see that the troughs



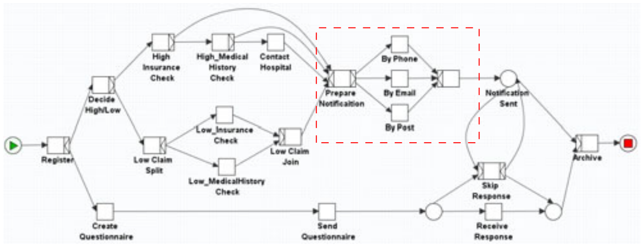
a. Model, M_1



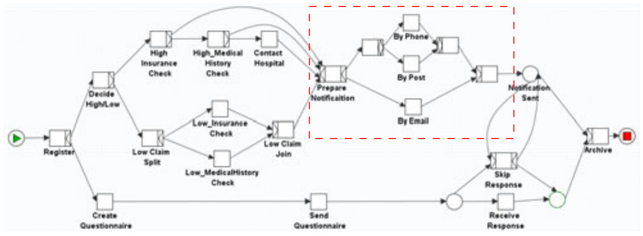
b. Model, M_2



c. Model, M_3



d. Model, M_4



e. Model, M_5

Fig. 4. Five variants of an insurance claim process of a travel agency represented in YAWL notation. The dashed rectangles indicate the regions of change from its previous model.

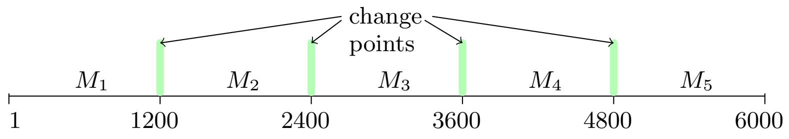


Fig. 5. Event log with traces from each of the five models juxtaposed. Also indicated are change points between models.

are observed around indices 24, 72 and 96 which are indeed the points of change (remember that we have split the log into 120 sub-logs with the change points at indices 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 due to the fact that the relation type counts would be alike for logs generated from these two process variants.

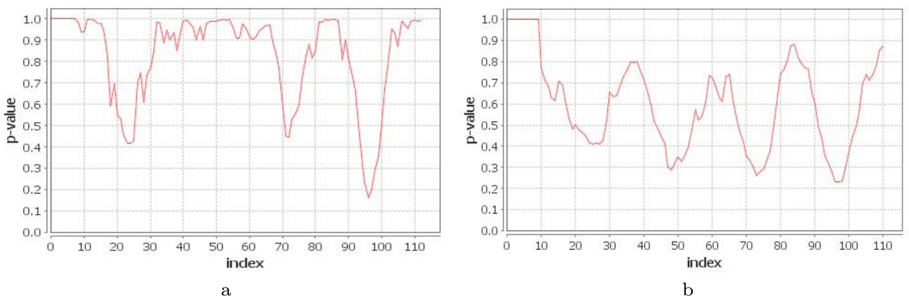


Fig. 6. (a) Significance probability of Hotelling T^2 test on relation counts (b) Average significance probability (over all activity pairs) of *KS*-test on *J*-measure. The event log is split into sub-logs of 50 traces each. *X*-axis represents the sub-log index. *Y*-axis represents the significance probability of the test. Troughs signify change points.

We have computed the *J*-measure for each sub-log and for every pair of activities, `a`, `b` in Σ ($\mathtt{a}F\mathtt{b}$, `b` follows `a` within a window of size 10). The univariate Kolmogorov-Smirnov test using a window size of $w = 10$ is applied on the *J*-measure of each activity pair. Fig. 6b depicts the average significance probability of *KS*-test on all activity pairs. It could be seen that significant troughs are formed at indices 24, 48, 72 and 96 which correspond to the actual change points. Unlike the relation type count 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). However in M_3 at least one of the modes need to be executed (OR construct). This results in a difference in the distribution of activity probabilities and their relationship probabilities which is elegantly captured in the *J*-measure.

We have considered the *J*-measure for each trace separately instead of at the sub-log level. Each activity pair generates a vector of dimension 6000 corresponding to the *J*-measure of that activity pair in each trace. The univariate

Kolmogorov-Smirnov test using a window size of $w = 400$ is applied to the vector corresponding to each activity pair in $\Sigma \times \Sigma$. Fig. 7 depicts the average significance probability of KS-test on all activity pairs. It could be seen that significant troughs are formed at indices 1200, 2400, 3600 and 4800. These are indeed the points where the models have been changed. Thus the features and approach proposed in this paper are shown to have significant promise in accurately identifying the points of change.

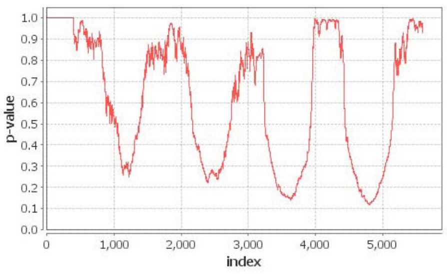


Fig. 7. Average significance probability (over all activity pairs) of KS-test on J-measure estimated for each trace. X-axis represents the trace index. Y-axis represents the significance probability of the test. Troughs signify change points.

The second objective in handling concept drift is that of *change localization*. In order to localize the changes (identify the regions of change), we need to consider each activity pair individually or a subset 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' due to the fact that 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 window count (WC) feature for the activity relation 'Contact Hospital' follows 'Register' on a window size of 10 in each trace separately. Fig. 8a depicts the significance probability of the univariate KS-test using a window size of $w = 200$ on this feature. It could be seen that one dominant trough is formed at index 1200 indicating that there exists a change in the region between 'Register' and 'Contact Hospital'. No subsequent changes with respect to this activity pair is noticed which is indeed the case in the models.

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 window count feature on the activity relations 'Send Notification By Phone' follows 'Prepare Notification', 'Send Notification By email' follows 'Prepare Notification' and 'Send Notification

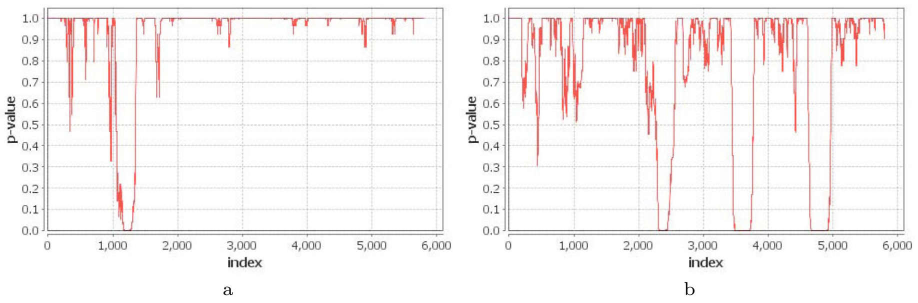


Fig. 8. (a) Significance probability of KS-test on WC feature estimated for the relation, 'Contact Hospital' follows 'Register'. Trough indicate change point w.r.t this feature. (b) Average significance probability (over activity pairs) of KS-test on WC feature estimated for the various modes of 'Send Notification' follows 'Prepare Notification' relation. Troughs indicate change point w.r.t these activities. X-axis represents the trace index. Y-axis represents the significance probability of the test.

By Post' follows 'Prepare Notification'. Fig. 8b depicts the average significance probability of the univariate KS-tests using a window size of $w = 200$ on the WC feature of these three activity pairs. We see three dominant troughs at around indices 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 a significance probability less than a threshold, δ. In this fashion, by considering activities (and/or activity pairs) of interest, one can localize the regions of change. Furthermore, one can also get answers to diagnostic questions such as "*Is there a change with respect to activity* a *in the process at time period t*"?

6 Outlook

Dealing with concept drifts raises a number of scientific and practical challenges. In this section, we highlight some of these challenges.

- *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 is needed. For example, if we would like to detect changes pertaining to a loop construct (insertion/removal/modification of loops as changes in process variants), tandem arrays [16] would be an appropriate feature to consider.
- *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 $3|\Sigma|$ features whereas the window count and J-measure generate $|\Sigma|^2$ features (corresponding to all activity pairs). On the one hand, such high

dimensionality makes the computational complexity intractable for most real life logs. On the other hand, changes being typically concentrated in a small region of a process makes it unnecessary to consider all features. There is a need for dimensionality reduction techniques that can efficiently select the most appropriate features.

– *Holistic Approaches:* In this paper, we discussed ideas on change detection and localization in the context of sudden drifts and owing to the control-flow perspective of a process. However, as mentioned in Section 3, data and resource perspectives are also equally important. So are the contexts of gradual, recurring and incremental drifts. 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.
– *Techniques for Drift Detection:* In this paper, we explored just the Hotelling T^2 test to deal with multi-variate data. In addition, we have dealt with multiple features by considering univariate hypothesis tests on each feature separately and averaging the test results over all features. Further investigation needs to be done on hypothesis tests devised naturally for multi-variate data. Also, determining an appropriate size of the window for hypothesis tests is nontrivial; this mandates further study on understanding the influence of window size on the results. Alternatives to hypothesis testing that can uncover drifts and diagnose the changes are a welcome addition to the repertoire of techniques for handing concept drifts in process mining.
– *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 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 Conclusions

This paper 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. The approach has been implemented in ProM and evaluated using synthetic data. This is a first step in the direction of dealing with changes in any process monitoring and analysis efforts. We considered changes only with respect to the control-flow perspective manifested as sudden drifts. However, there is much to be done on various other perspectives mentioned in this paper. Moreover, to further validate the approach we plan to conduct extensive case studies based on real-life event logs.

Acknowledgments. R.P.J.C. Bose and W.M.P. van der Aalst are grateful to Philips Healthcare for funding the research in process mining.

References

1. Žliobaitė, I.: Learning under Concept Drift: an Overview. Technical report, Faculty of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania (2009)
2. Pechenizkiy, M., Bakker, J., Žliobaitė, I., Ivannikov, A., Kärkkäinen, T.: Online Mass Flow Prediction in CFB Boilers with Explicit Detection of Sudden Concept Drift. SIGKDD Explorations 11(2), 109–116 (2009)
3. Tsymbal, A., Pechenizkiy, M., Cunningham, P., Puuronen, S.: Handling Local Concept Drift with Dynamic Integration of Classifiers: Domain of Antibiotic Resistance in Nosocomial Infections. In: CBMS, pp. 679–684 (2006)
4. Weber, B., Rinderle, S., Reichert, M.: Change Patterns and Change Support Features in Process-Aware Information Systems. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 574–588. Springer, Heidelberg (2007)
5. Mulyar, N.: Patterns for Process-Aware Information Systems: An Approach Based on Colored Petri Nets. PhD thesis, University of Technology, Eindhoven (2009)
6. Schonenberg, H., Mans, R., Russell, N., Mulyar, N., van der Aalst, W.M.P.: Process Flexibility: A Survey of Contemporary Approaches. In: Dietz, J., Albani, A., Barjis, J. (eds.) Advances in Enterprise Engineering I. LNBIP, vol. 10, pp. 16–30. Springer, Berlin (2008)
7. Regev, G., Soffer, P., Schmidt, R.: Taxonomy of Flexibility in Business Processes. In: Proceedings of the 7th Workshop on Business Process Modelling, Development and Support, BPMDS, Citeseer (2006)
8. Ploesser, K., Recker, J.C., Rosemann, M.: Towards a Classification and Lifecycle of Business Process Change. In: Proceedings of BPMDS, vol. 8 (2008)
9. Günther, C.W., Rinderle-Ma, S., Reichert, M., van der Aalst, W.M.P.: Using Process Mining to Learn from Process Changes in Evolutionary Systems. International Journal of Business Process Integration and Management 3(1), 61–78 (2008)
10. Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. Machine learning 23(1), 69–101 (1996)
11. Smyth, P., Goodman, R.M.: Rule Induction Using Information Theory. In: Knowledge Discovery in Databases, pp. 159–176. AAAI Press, Menlo Park (1991)
12. Blachman, N.M.: The Amount of Information that y Gives About X. IEEE Transactions on Information Theory IT-14(1), 27–31 (1968)
13. Sheskin, D.: Handbook of Parametric and Nonparametric Statistical Procedures. Chapman & Hall/CRC (2004)
14. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. Information Systems 30(4), 245–275 (2005)
15. Vinter Ratzer, A., Wells, L., Lassen, H.M., Laursen, M., Qvortrup, J.F., Stissing, M.S., Westergaard, M., Christensen, S., Jensen, K.: CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. In: van der Aalst, W.M.P., Best, E. (eds.) ICATPN 2003. LNCS, vol. 2679, pp. 450–462. Springer, Heidelberg (2003)
16. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Abstractions in Process Mining: A Taxonomy of Patterns. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 159–175. Springer, Heidelberg (2009)