Comparing Concept Drift Detection with Process Mining Tools

Nicolas Jashchenko Omori
State University of Londrina (UEL)
Londrina, Brazil
omori@uel.br

Gabriel Marques Tavares
State University of Londrina (UEL)
Londrina, Brazil
gtavares@uel.br

Paolo Ceravolo
Università degli Studi di Milano (UNIMI)
Milano, Italy
paolo.ceravolo@unimi.it

Sylvio Barbon Jr.
State University of Londrina (UEL)
Londrina, Brazil
barbon@uel.br

ABSTRACT

Organisations have seen a rise in the volume of data corresponding to business processes being recorded. Handling process data is a meaningful way to extract relevant information from business processes with impact on the company's values. Nonetheless, business processes are subject to changes during their executions, adding complexity to their analysis. This paper aims at evaluating currently available Process Mining tools that handle concept drifts, i.e. changes over time of the statistical properties of the events occurring in a process. We provide an in-depth analysis of these tools briefly comparing their differences, advantages, and disadvantages.

CCS CONCEPTS

• **Applied computing** → **Business process management**; **Business process modeling**; • **Information systems** → *Data stream mining*;

KEYWORDS

Process Mining, Online, Concept Drift

ACM Reference format:
Nicolas Jashchenko Omori, Gabriel Marques Tavares, Paolo Ceravolo, and Sylvio Barbon Jr.. 2019. Comparing Concept Drift Detection with Process Mining Tools. In *Proceedings of XV Brazilian Symposium on Information Systems, Aracaju, Brazil, May 20–24, 2019 (SBSI'19),* 8 pages.
https://doi.org/10.1145/3330204.3330240

1 INTRODUCTION

Concerned about their success, organisations are interested in having precise control over their processes and rapidly reacting to relevant events recorded in their information systems during process execution. Data stream analysis may offer new opportunities for these organisations but at the same time impose new challenges [12]. Among them, Concept Drift (CD) detection is crucial, as it identifies if the patterns followed by data are changing and, by consequence, if the models adopted to interpret them stay valid or require an update [8].

Recently, process mining (PM) techniques have arisen as a valuable tool to interpret business process data generated by organisations. PM uses process modelling and analysis as well as notions from data mining (DM) and machine learning (ML) [26]. In traditional PM techniques, one has access to data from event logs recorded by systems controlling the execution of processes [20]. Thus, models are constructed based on historical series. However, CD in PM may be as common as in other areas, giving the fact that the entire historical series may not be appropriately significant of a running execution [19]. Identifying a drift, a point in time where there is a statistical difference between the process behaviour before and after the said point is then decisive to guide the update of models [14].

In recent years the implementation of CD techniques for PM has received attention. Particularly, two algorithms [4, 15] for concept drift detection are widely available on popular PM software.

The first method was proposed by Bose et al. [4] and a plugin, named **Concept Drift**, implements it for the Process Mining Workbench (ProM) package manager. The method consists in extracting and selecting the features of the event log, generating a set of control populations used for comparisons. Then, displaying an interactive visualisation of the drifts detected.

The second method, **ProDrift**, is proposed by Ostovar et al. [15] and is provided for the Advanced Process Analytics Platform (Apromore). The method uses two adjacent adaptive windows and performs statistical tests over distributions of behavioural relationships between events.

Bose et al. [4] and Ostovar et al. [15] techniques focus on the same goal: detecting CD. However, each algorithm carries on particularities such as hyper parametrisation, visualisation capabilities, and other specific functions. Selecting the most suitable solution is not an easy task, as well as configuring the hyperparameters to support desirable CD detection. This main hindrance is related to the different behaviour of each real-life process which requires ad-hoc settings [10]. In other words, the demand for a PM tool capable of supporting accurate CD analysis with a straightforward setup process and offering human-friendly interpretations is increasing.

The goal of this work is to evaluate available process drift solutions using a real-life event log, offering an in-depth analysis of such methods based on sensitivity, hyperparameters dependence and software interface. More specifically, we compared the ProM's

Concept Drift plugin and ProDrift from Apromore over a dataset of a Process Control System composed of four years of event logs.

The rest of this paper attends the following organisation: Section 2 revises the main concepts about process mining. Section 3 presents an in-depth overview of the Process Control System event log, the software used in the experiments and our evaluation criteria. Section 4 discusses the obtained results and compare them over the dataset experiment. Lastly, Section 5 concludes the paper, emphasising open avenues for future work.

2 PROCESS MINING

In our globalised world, new technologies emerge at high rates and the production of data has been in a constant rise. A significant amount of the produced data regards business processes executions recorded as event logs. To tie organisations needs and process data, PM offers a set of techniques that retrieves information from event logs and gives companies a better understanding of their processes, further supporting business decisions by making clear how the processes are developing according to event log data.

This relatively new field merges knowledge from business process management (BPM), which studies operational business processes from the information technology and management sciences standpoint, with data mining and machine learning (ML), which focus on data analysis [26]. PM main focus is to discover, monitor and enhance business processes towards a clear understanding of the process [25].

Traditional process discovery techniques aim at extracting an abstract representation of an event log, i.e. a process model, that best describes the recorded behaviour [7]. There are several notations which serve as models, and it is important to notice that there is no perfect model representation, meaning that there is a wide variety of algorithms (alpha-algorithm [23], the inductive miner [13], the heuristic miner [28], among others) for model extraction and the final model may be different within different discovery techniques.

Monitoring processes is commonly referred to as conformance checking techniques, which aims at detecting inconsistencies amongst a process model and an event log corresponding to the same process [18]. Moreover, conformance may provide a means for quantifying the deviations, posing as an essential tool for noise identification. Finally, process enhancement uses previous analysis as the basis for process improvement by changing or extending the original model [1]. An enhancement technique may either repair the model by updating its relations or extend the model by adding new information, such as timestamps [26].

Table 1 shows an example of an event log. Each row of the table represents one *event*, which is an execution of an *activity* at a certain *time*. Moreover, each activity corresponds to a specific *case*, where a case is an instance of process execution. Furthermore, a *trace* is a sequence of activities from the same case, implying that different cases may have the same trace. We can, then, infer that cases 3 and 5 are executions of the same process even though they are distinct instances and may run differently. From Table 1, we can conclude that cases 5 and 7 have the same trace and that the group of cases 1, 3, 5 and 7 is a set of recorded events generated by the same business process.

For event log processing, it is expected that the log is time ordered, respecting the real sequence of events. Since an event is a recording of an activity belonging to a process instance, the required attributes of an event are the case identification, an activity name, and a timestamp, as seen in Table 1.

Table 1: Event log example

Case ID	Activity	Timestamp
Case 5	Solicitação	2018/02/09 11:00:00
Case 7	Solicitação	2018/02/09 11:45:20
Case 1	Solicitação	2018/02/09 11:55:47
Case 7	Autorização	2018/02/10 16:27:36
Case 5	Autorização	2018/02/10 16:27:45
Case 5	Distr. Diretoria	2018/02/10 17:13:27
Case 1	Distr. Setor	2018/02/10 17:13:56
Case 3	Distr. Diretoria	2018/02/13 12:00:50
Case 3	Em Execução	2018/02/17 09:10:20
Case 3	Cancelamento	2018/02/17 10:30:00
Case 7	Distr. Diretoria	2018/02/20 11:00:00
Case 1	Conclusão	2018/02/20 17:00:00

Definition 1 (*Event, attribute, trace* [26]). Let Σ be the *event universe*, i.e., the set of all possible event identifiers. Events may have various *attributes*, such as timestamp, activity, resource, associated cost, among others. Let AN be the set of attribute names. For any event $e \in \Sigma$ and name $n \in AN$, then $\#_n(e)$ is the value of attribute n for event e, if event e has an attribute n, else $\#_n(e)$ is null. A *trace* is a non-empty sequence of events $\sigma \in \Sigma^*$ where each event appears only once and time is non-decreasing, i.e., for $1 \leq i < j \leq |\sigma|$: $\sigma(i) \neq \sigma(j)$.

Definition 2 (*Case, event log* [22]). Let C be the *case universe*, that is, the set of all possible case identifiers. An *event log* is a set of cases $L \subseteq C$ where each event appears only once in the log, i.e., for any two different cases, the intersection of their set of events is empty.

In this work, Causal nets (C-net) are used as the process modelling notation as they offer a human-readable representation of the process as well as they are often used in PM works.

Definition 3 (*Causal net* [21]). A *Causal Net* is a tuple $C = (A, a_i, a_o, D, I, O)$ where A is a finite set of activities, a_i is the start activity, a_o is the end activity, $D \subseteq A \times A$ is the dependency relation, $AS = \{X \subseteq \mathcal{P}(A) | X = \{\emptyset\} \vee \emptyset \notin X\}$, $I \in A \rightarrow AS$ defines the set of possible input bindings per activity and $O \in A \rightarrow AS$ defines the set of possible output bindings per activity.

2.1 Current Issues in PM

Previously, we have discussed traditional PM fundamentals and techniques which are applied in scenarios where one has access to all event log data produced by a process that has already run for some period. The consequences are that traditional algorithms deal with an event log with complete cases, that is, cases that went through its final activity.

In real life environments though, organisations interests are focused on the instant feedback of their processes since old event logs are mostly deprecated versions of the current process and not capable of representing the current state. Moreover, traditional PM methods are not suitable solutions in environments where processing time is limited, immediate responses to anomaly detection are required, and event logs are too large. Thus, the need for on the fly analysis is rising, making event stream solutions more relevant today [3].

One of the implications when dealing with data streams is that the stream is potentially infinite, consequently limiting the usage of processing resources, such as memory and time [8]. Moreover, the assumption of all training data being available to create a model is not valid, that is, different time periods may imply different distributions of data, requiring the learning system adaptation. However, data stream mining solutions cannot be directly applied in event streams since there is a mismatch at the representation level, where stream analysis is usually set at the tuple level while PM is set at the business case level, i.e., multiple recorded events compose a case, representing the sequence of activities in a process instance.

Definition 4 (*Stream* [12]). A *stream* $\mathcal{S}$ is defined in the format $\mathcal{S} = \{i_1, i_2, i_3, ..., i_n, ...\}$, where i corresponds to a pair $P(\vec{x}, y)$ when the ground truth for that instance is known or simply $\vec{x}$ when it is not, with $\vec{x}$ being the feature vector of that instance and y being its label, and n is possibly infinite.

Additionally, learning from data streams requires continuous adaptation since data behaviour can change over time, producing different data distributions [8, 12]. This phenomenon is known as concept drift, and its occurrence is common in process environments where organisational changes and hidden contexts influence the process execution, as underlined in the Process Mining Manifesto [24]. The utmost consequence of drift is an outdated model, incapacitating its capability of recognising the new behaviour.

Maaradji et al. [14] define process drift as a point in time where there is a statistical difference between the process behaviour before and after the said point. According to Seelinger et al. [19], a significant behavioural change of the process execution over time is characterised as process drift, exemplifying that almost all traces after a process change follow the new data distribution. Towards adaptation, the model must use newly observed data to update its representation always balancing the influence of old and new data.

Definition 5 (*Concept drift* [8, 12]). Given the sequence of streams $\langle \mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_i, ... \rangle$ where $\mathcal{S}_i$ is a set of examples generated by some distribution $\mathcal{D}_i$. Given $P^t(\vec{x}, y)$, at each timestamp t, the feature vector $\vec{x}^t$ corresponds to a class y^t. Given two distinct points in time t and $t + \Delta$, given $\mathcal{D}^t \neq \mathcal{D}^{t+\Delta}$, if there is a $\vec{x}$ that satisfies $P^t(\vec{x}, y) \neq P^{t+\Delta}(\vec{x}, y)$, then a *concept drift* has happened.

Bose et al. [5] further distinguish two types of drift, online and offline. The first refers to the real-time identification of drift while the later indicates scenarios where drift is detected after a process has ended.

2.1.1 *ProM's Concept Drift plugin.*

Bose et al. [4] method, available in ProM's Concept Drift plugin, is grounded on the premise that event logs can be characterised by the relationship between activities, mainly the dependencies are used in the feature extraction and selection part of the framework. The dependencies can be explained as a follows (or precedes) relation, such as, for a pair of activities *a, b* $\in L$ can be determined if either *a always, never* or *sometimes* follows (or precedes) *b*. Event log features are then defined based on these relationships. There are four types of features proposed, Relation Type Count, Relation Entropy, Window Count and J-measure, being the first two global and the two latter being local features. After that, the features are used to transform the event log in a data stream, which in itself is used to define the sample populations that will be compared using the statistical tests. The populations are generated using sliding windows, with the windows either having fixed or adaptive size, being continuous or non-continuous (there can be a gap between populations), and the windows can or cannot overlap. Then, the populations are compared using one of the three available statistical tests: Kolmogorov-Smirnov test, Mann-Whitney test, and the Hotelling T^2 test, each of those having different meanings for deciding if two populations differ from each other. At this step, concept drift detection is performed, and the next steps are done to provide an analyst with an intuitive visualisation of the significance probabilities as a drift plot.

2.1.2 *Apromore ProDrift plugin.*

Ostovar et al. [15] implementation, in Apromore, assume that a business process drift detection may identify a time point before and after which there is a statistically significant difference between the observed process behaviour. With that, they propose to represent the process behaviour using the $\alpha+$ relations [2], which are a set of rules to represent different relations between activities. Thus, each sliding window contains events creating a sub-log, from which the $\alpha+$ relations and their frequencies are extracted, building a so-called contingency matrix. When a new event arrives, the G-test of independence [9] is applied in the contingency matrices, and if the significance probability is below a defined threshold, then the $\alpha+$ relations in the windows come from different distributions, meaning a process drift has occurred.

3 MATERIALS AND METHODS

3.1 Process Control System Event Log

Process Control System is concerned with monitoring the progress of an order, consisting of the three applications: feedback control, in-process control and feed-forward control [6]. Process Control System often exhibit regular and predictable events, but some dynamic changes in improving the quality of services delivered are required. In this scenario, dealing with CD in Process Control System demand attention, mainly in the detection of drift points to clarify some aspects such as novel procedures, security politics and service portfolio expansion.

Motivated by these facts, as a case study to compare CD detection techniques, an event log with recorded activities from the process control system of a software house was extracted. The extracted business process comes from the Department of Information and Communications Technology (DICT) from a public university, whose role is to support the university providing software solutions and the infrastructure for them.

The DICT event log records the execution of inner processes as services are requested. Those services may vary depending on the applicant's goal, that is, an applicant may ask for a software solution for their department or a correction of an internet node, for instance. Independently to the requested service, the DICT has a set of activities which are used to process the request. Figure 1 shows the resulting Causal net by applying the Data-aware Heuristic Miner with standard parameters. The extracted event log contains process instances from January 2014 through August 2018, and the Causal net represents all the process behaviour in this period.

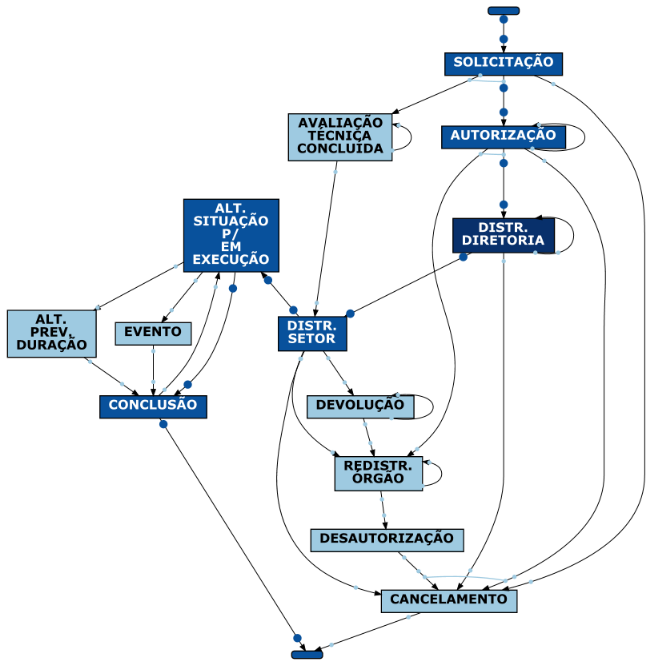


Figure 1: Causal net that represents the event log

All traces in the event log start with activity SOLICITAÇÃO, which represents the request of a service, i.e., the starting point of every process instance. Also, all traces end in either CONCLUSÃO or CANCELAMENTO, which mean the service was concluded or cancelled, respectively. Therefore, we can assume that all processes start with a request which is later either responded or cancelled.

Table 2 shows the number of appearances of each activity through the log. The most evident pattern is a group of activities occurring over ten thousand times, thus showing the most common flow of execution in DICT. First, a request is registered, then authorised and distributed for production, after that, the request is executed and finally concluded. This set of activities is connected in a standard flow inside the organisation. However, a DICT business process might run into unforeseen scenarios, which results in the cancelling of a request. This phenomenon is represented in the next group of activities (e.g. CANCELAMENTO, DESAUTORIZAÇÃO, DEVOLUÇÃO), which is less common but as relevant as the previous group. Ultimately, the most uncommon activities are shown in Table's 2 tail, representing rare cases where the process flow is odd.

As an extension of the previous analysis, we have selected the two most common traces in the event log. The trace ⟨SOLICITAÇÃO, AUTORIZAÇÃO, DISTR. DIRETORIA, DISTR. SETOR, ALT. SITUAÇÃO P/ EM EXECUÇÃO, CONCLUSÃO⟩ occurs 9308 times, representing the regular execution flow of answered requests. The second most occurring one is ⟨SOLICITAÇÃO, AUTORIZAÇÃO, DISTR. DIRETORIA, DISTR. SETOR, CANCELAMENTO⟩, which occurs 166, representing cancelled traces. It can be inducted that cancelled requests are a minority; moreover, the cancelling pattern may vary more commonly than the concluded one, which shows that a cancelled request is a representation of an unusual request.

Process statistics can be further examined for a complete view of the process; thus Table 3 shows several metrics regarding case, trace and time characteristics. Upon initial inspection, the number of mean cases per day (24.49) manifests a busy process for a medium-sized organisation (around forty people are involved in the process). Complementarily, the mean number of events is also affected by the high number of requests every day. Regarding trace length, the mean is around 6 since the majority of cases have either 5, 6 or 7 activities for both concluded and cancelled requests. Finally, the mean duration is around three weeks, which is reasonable in a university environment with different inner organisations involved in the business process, raising the processing time.

Table 2: Occurrence count in DICT event log from January 2014 to August 2018

Activity	Number of Occurrences
DISTR. DIRETORIA	10701
AUTORIZAÇÃO	10558
DISTR. SETOR	10527
SOLICITAÇÃO	10434
CONCLUSÃO	10014
ALT. SITUAÇÃO P/ EM EXECUÇÃO	10013
CANCELAMENTO	439
REDISTR. ÓRGÃO	381
DESAUTORIZAÇÃO	229
DEVOLUÇÃO	176
ALT. PREV. DURAÇÃO	65
AVALIAÇÃO TÉCNICA CONCLUÍDA	17
EVENTO	12
ALT. SITUAÇÃO P/ SUSPENSO	8
AGUARDANDO AVALIAÇÃO TÉCNICA	4
ALT. SITUAÇÃO P/ AG. MATERIAL	1
ALT. SITUAÇÃO P/ AG. SUBSOLIC	1

Table 3: Statistics from DICT event log

Statistics	Value
Total cases	10430
Mean cases per day	24.49
Max cases per day	103
Total events	63580
Mean events per day	57.85
Max events per day	256
Mean trace length	6.09
Max trace length	40
Mean case duration (in days)	22.84

The selection of this dataset was made on behalf of changing the behaviour of its execution. Regulated by laws that influence financial budget affecting the service activities, and frequent new service politics owing to improve its quality (mainly in IT perspectives) the DICT present several CD to be analysed.

3.2 Process Mining Softwares

Despite being a recent field of research, process mining has already shown a wide range of techniques during recent years. However, access to these proposed techniques is not always a simple task. Thus, a necessity arises for a framework that integrates various techniques and methods practically. Inspired by that, Dongen et al. [27] proposed ProM, an open-source, extensible, process mining framework, with the objective of providing a set of techniques for exploring event logs. ProM is widely used in the process mining community, as evidenced by a number of publications [11, 16], and there is a lot of support from the community on their site's discussion forum. The various techniques available in ProM are distributed in plugins, which are available at the ProM package manager or in *jar packages* that can be installed manually.

Also inspired by the need of PM framework, Apromore[1] was launched firstly as an advanced process model repository [17] but rapidly became an open-source business process analytics platform that merges state-of-the-art process mining research with the management of process model collections, serving both academic and enterprises needs. As well as ProM, Apromore offers a varied set of PM techniques ranging from automated process discovery to process prediction and drift detection. The currently employed process drift detection technique [14, 15], known as ProDrift version 4.5, is also available as a standalone tool, facilitating the use outside of Apromore's interface. Apromore is licensed under LGPL version 3.0, and its source code is available publicly[2].

ProDrift provides the detection and characterisation of process drifts accepting as input an event log in the XES or MXML format. Furthermore, ProDrift considers two types of stream processing: event-based and trace-based. The first one handles the stream consuming one event at a time, which is more relatable to real environments and is considered an online solution for drift detection. On the other hand, the trace-based solution groups the cases before processing and creates a stream of traces, such technique is considered an offline one since it only deals with complete event logs.

For drift detection, statistical tests are performed over the distributions of process runs (when dealing with a stream of traces) or $\alpha+$ relations (when dealing with a stream of events). Both detection types are capable of handling sudden drifts, i.e., drifts where the concept change is abrupt along the stream, consequently meaning that there is one specific drift point. Moreover, the trace stream method also deals with gradual drifts, i.e. drifts where a new concept appears rather slowly but becomes more frequent along the stream while the initial concept fades away.

Beyond ProM and Apromore, there is some other software that employs process mining techniques, such as Disco[3], Minit[4] and QPR[5], all of those being commercial solutions. In this research, only ProM and Apromore were used, as the concept drift detection capabilities are not present on the other software. Moreover, when searching the strings "ProM process mining" and "Apromore" in the Google Scholar database, 67.600 and 419 results are retrieved, respectively, showing the high relevance of those PM frameworks in the community.

[1]http://apromore.org/
[2]https://github.com/apromore/ApromoreCode
[3]https://fluxicon.com/disco/
[4]https://www.minit.io/

3.3 Comparison

Both drifting detection methods provide a diverse set of parameters, and varying those configurations may impact directly in the drifting results. Regarding ProDrift, we have conducted experiments considering the event-based method since it offers an online solution by handling a stream of events, approximating itself to real scenarios where each event is recorded at a time. The following parameters and values assumed for testing were:

- Drift detection mechanism: *events*;
- Window size: an integer representing the window size. We have used 58, 263, 1135 and 2270. The reason behind is that we computed the mean number of DICT events per day, week, month and bimester, representing cycles within the organisation. Thus, the method can provide feedback on the process depending on the stakeholder's needs;
- Fixed window mode: controls if the window size can or cannot change throughout the stream. For each window size according to the previous parameter, a mode with fixed window size was also applied;
- Noise filter threshold: specifies the noise threshold to filter noisy $\alpha+$ relations and ranges from 0 to 1. ProDrift's documentation recommends 0 for artificial datasets and 1 for real ones. For our experiments, the noise threshold was set to 0.0, 0.1, 0.2 and 0.3;
- Drift detection sensitivity: specifies how sensible the algorithm is to drift. Since the main focus of this research is to experiment with drift, we have used all possible configurations in this parameter, which are *verylow*, *low*, *medium*, *high* and *veryhigh*.

The parameters values described previously were applied in a grid methodology, i.e. all possible arrangements between said parameters were tested, resulting in 160 different tests. Thus, by extensively exploring the method we can later infer what parameters influence in the results.

Regarding ProM's Concept Drift plugin, there is a variety of parameters that modify the behaviour of the drift detection. Only the local features were used, as the options using the global features did not work properly, i.e. crashed. Therefore, the parameters and values used for testing were as follows:

- Log Configuration Step: defines if the log will be split or not. We ran without splitting and with a split every 100 instances;
- Feature Scope: selects the scope of global and local features. Only the local features were used, with all the activities selected;
- Feature Type: chooses if the features will be based on follows, precedes or both relations. All three options were used;

[5]https://www.qpr.com/solutions/process-mining

- Relations Counts: There is only one option, Numeric Value;
- Metric Type: the choice of the two local features, Window Count and J-Measure, is available. Both types of metrics were used;
- Drift Detection Method: Three methods for drift detection are available (Gamma Test is shown, but it cannot be selected). Tests were run with Kolmogorov-Smirnov and Mann-Whittney options;
- Population Options: generates the populations that will be compared. There are three pairs of options, Fixed-Window or ADWIN (Adaptive Windows), Sudden or Gradual drift search and Trace Amount or Time Periods. The first pair defines if the window will be of fixed or adaptive size. The second pair defines if the type of drift detected will be a Sudden Drift or a Gradual Drift. The last pair defines the way that the window size will be calculated, in regards to the amount of traces or in regards to a time period.

The same grid methodology was used, resulting in 108 different tests, as some pairs of options are invalid, i.e. Fixed-Sized populations cannot be selected along the time period option.

There is a need for measuring the drifts found by the algorithms. Thus, in order to check if the drift points are real, we applied a process discovery algorithm (discover graph) both before and after the drift point. This way, we can compare graphs of the process and check if its behaviour is affected by the drift. When the process graphs before and after the drift are equal, then a drift has not occurred, and the method indicated a false drift point.

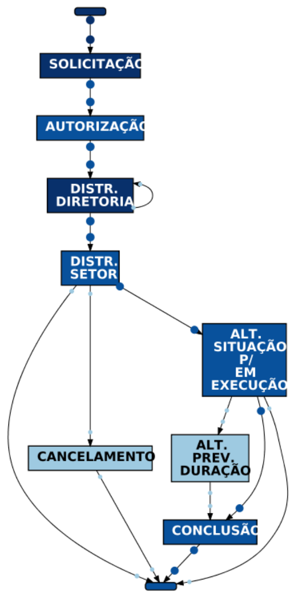
Moreover, to add on the evaluation, the number of detected drifts is tracked as much as the number of unique drifts. Thus, the precision of the methods can be measured and their sensitivity compared. Furthermore, their interface is evaluated, showing their usability. Finally, a table is presented summarising both methods and highlighting their differences.

4 RESULTS AND DISCUSSION

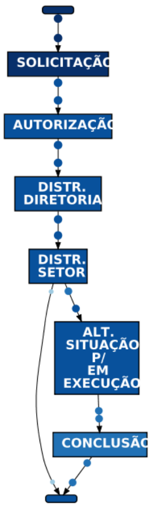
4.1 Concept Drift Sensitivity

For the 160 different tests submitted to ProDrift, a total of 5182 drift points were found, being 678 of them unique. Considering that the event log contains 63580 events and 10430 cases, the number of unique drift points is reasonable. Moreover, many of the tested configurations are very sensible, thus the high number of found drifts. However, 40 tests found no drift points in the event log, showing that the parameters configuration directly affects the results of the algorithm.

The most common drift point, found in 35 different tests, was the event 6607, which happened on May 22, 2014. To verify this drift's veracity, we divided the event log before and after said point and submitted to a traditional process discovery algorithm, the Interactive Data-aware Heuristic Miner, which is available as a plugin in ProM. The output of this discovery algorithm is in the form of a Causal net. Figures 2a and 2b show the Causal nets before and after the found drift point. It is clear that the Causal net from Figure 2a is more complex than the one in Figure 2b, presenting more arcs and connections within activities. One example is that both `CANCELAMENTO` and `ALT. PREV. DURAÇÃO` activities are present in the first Causal net and not in the latter. From a PM perspective, we can interpret both models as different processes since they are composed of a different set of activities and arcs. Thus, this can be considered a drift point since the behaviour observed before and after the point is different, meaning the process has changed during its execution.



(a) Before drift point



(b) After drift point

Figure 2: Process model before (a) and after (b) found drift point on May 22: `CANCELAMENTO` and `ALT. PREV. DURAÇÃO` did not occur after the CD point.

As of ProM's Concept Drift plugin, we used the same testing method and, for the 104 tested configurations, the 42 configurations with detected drifts have found 1317 drift points, a mean of 11.41 drift points per configuration. Of the total 1317 points, only 187 points are unique (14% of the total), indicating consistency in the points that are detected as drifts.

4.2 Hyperparameters dependence

Table 4 presents the hyperparameters related to each tool used. By comparing the two approaches, ProM contains more hyperparameters and options within them. Having more parameters enables numerous configurations of setups to explore the event log more in-depth. However, a high number of parameters makes the approach more complex, decreasing its usability. Thus, ProDrift's approach is more user-friendly parameter wise, making it easier for non-experts to use the tool.

In regards to ProM's hyperparameters, the drift detection method strongly affects the drift detection sensitivity, as tests using the Kolmogorov-Smirnov drift detect an average of 1 drift point per configuration while tests with Mann-Whittney drift detection algorithm average 21.83 drifts per configuration. Other hyperparameters, such as choosing between Window Count or J-Measure, did not affect the number and locations of the found drift points, with other settings being equal.

Table 4: Comparison Summary

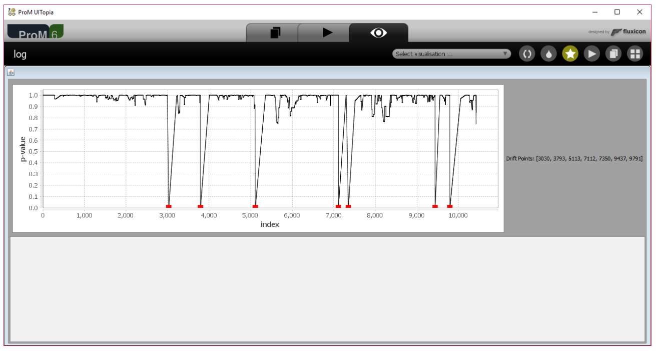
Characteristic	ProM	ProDrift
Detection Method	Kolmogorov-Smirnov Test, Mann-Whittney Test or t-Test	G-Test
Windowing	Fixed Size and Adaptive Window, parameters: Trace amount or Time Periods, Pop Size, Step Size, Min size, Max Size	Fixed Size or Adaptive Window, parameter: Window Size
Feature Selection	J-measure, a pair of activities and Window Count metric types, parameters: Feature type, Relation Counts, Metric Type	$\alpha+$, parameters: none.
Sensitivity	p-value, parameters: p-value threshold.	*Very Low*, *Low*, *Medium*, *High* and *Very High*, parameters: Cumulative change (%), Drift detection sensitivity.
Noise	-	Filtering, parameters: Relation noise filter (%)
Drift Characterization	Sudden Drifts and Gradual Drifts	All drift points, parameter: Drift Characterization (on or off)
Stream Type	Only a stream of traces can be used. Parameters: none	Both streams of events and traces can be used. Parameters: Trace-based (Runs) or Event-based ($\alpha+$)

For ProDrift, the most influential hyperparameter is window size. Setting up a small window highly increases the number of detected drift points since the sub log size is too small and any new deviation results in a detected drift. Higher window sizes make the detection less sensitive, once the populations are larger and more new behaviour must be observed to trigger a drift alert.

Furthermore, selecting the fixed window size hyperparameter plaster the algorithm since it disables its adaptation maneuver. The noise filter threshold depends on the characteristics of the event log since it filters away less frequent traces. It is recommended that for real event logs the noise should be set to 0.1 or 0.2 while for artificial event logs, the ideal value for noise is 0. The same goes for the drift detection sensitivity, lower sensitivities find more drifts and are recommended for real-life logs, and higher sensitivities trigger fewer drifts and are advised for artificial logs.

4.3 Software Interface

ProM's Concept Drift plugin is integrated into the ProM framework. As such, its entire operation is done through the ProM framework's graphical user interface. It uses an XLog as the input, which is pretty convenient, as ProM can import a variety of types of files, such as *CSV*, *XML*, *XES*, and convert them to an XLog. There are 4 windows of options that are used for setting the parameters for the plugin. Then, after running, the output is shown in Figure 3. The red dots in the graph indicates a drift point, and there is a list of drift points in the right side of the image. Although the drift points are shown in this screen, the dates in which the drift points have happened are outputted only on the system's standard output. Thus, the program must be run from the command line to retrieve the drift points. Though some ProM plugins can be used on the command line without interaction with the graphical user interface, the Concept Drift cannot. This way, it is not possible to run multiple tests in batches.



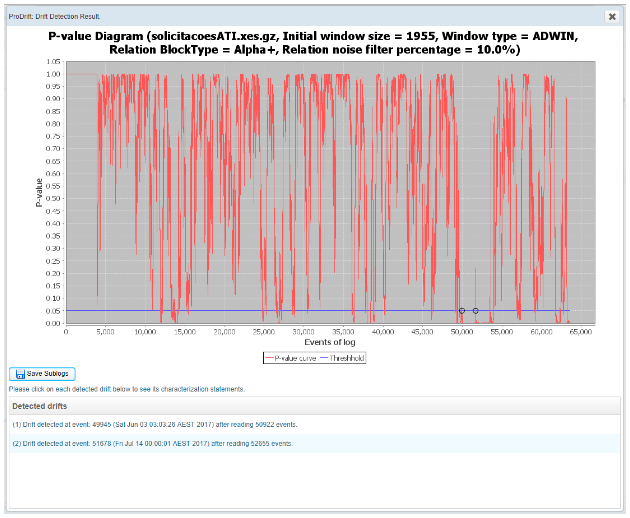
Drift Points: [3030, 3793, 5113, 7112, 7350, 9437, 9791]

Figure 3: ProM's Concept Drift graphical interface

Though being integrated into the Apromore framework, ProDrift is also available as a standalone tool that can be run from the command line. The Apromore framework is a web server that provides access to all of its tools, and to use it, one can deploy the server locally or access a remote server[6]. To use the ProDrift plugin, the event log have to be imported to the server in the following formats: *CSV*, *XML*, *MXML* or *XES*. There is only one page of configurations to be chosen, which is pretty straightforward. The output is then displayed as a graph that shows the p-values of the population comparisons and, below that, a list of drifts points found, as shown in Figure 4. Despite having a command line version, it is still difficult to automate the tests, as every test outputs to the

[6]http://apromore.org/platform/cloud

system's standard output and a window have to be manually closed every time. That being said, both tools provide a good and easy use of the graphical interface.


ProDrift: Drift Detection Result.

P-value Diagram (solicitacoesATI.xes.gz, Initial window size = 1955, Window type = ADWIN, Relation BlockType = Alpha+, Relation noise filter percentage = 10.0%)

Please click on each detected drift below to see its characterization statements.

Detected drifts

(1) Drift detected at event: 49945 (Sat Jun 03 03:03:26 AEST 2017) after reading 50922 events.

(2) Drift detected at event: 51678 (Fri Jul 14 00:00:01 AEST 2017) after reading 52655 events.

Figure 4: ProDrift output graphical interface

5 CONCLUSION

The amount of data that organisations have to handle nowadays is steadily growing. In addition to this growth, data is also constantly changing, which means organisation's processes have to keep changing themselves to adapt to the new requirements created by those changes.

This paper discusses and analyses the available tools that can help one detect and react to an unexpected change of behaviour in a certain process, by applying those tools in a real-life environment. In future works, we should try and propose new methods for concept drift detection, applying the knowledge acquired in this research.

6 ACKNOWLEDGEMENTS

We would like to thank Fundação Araucária for providing scholarships.

REFERENCES

[1] W. Aalst, van der, A. Adriansyah, and B. Dongen, van. Replaying history on process models for conformance checking and performance analysis. *WIREs Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.

[2] A. Alves De Medeiros, B. Dongen, van, W. Aalst, van der, and A. Weijters. *Process mining : extending the alpha-algorithm to mine short loops.* BETA publicatie : working papers. Technische Universiteit Eindhoven, 2004.

[3] S. Barbon Junior, G. M. Tavares, V. G. T. da Costa, P. Ceravolo, and E. Damiani. A framework for human-in-the-loop monitoring of concept-drift detection in event log stream. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 319–326, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[4] R. J. C. Bose. *Process mining in the large: preprocessing, discovery, and diagnostics.* PhD thesis, Eindhoven University of Technology, Eindhoven, 01 2012.

[5] R. P. J. C. Bose, W. M. P. van der Aalst, I. Žliobaitė, and M. Pechenizkiy. Dealing with concept drifts in process mining. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):154–171, Jan 2014.

[6] D. Braha. *Data mining for design and manufacturing: methods and applications*, volume 3. Springer Science & Business Media, 2013.

[7] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst. On the role of fitness, precision, generalization and simplicity in process discovery. In R. Meersman, H. Panetto, T. Dillon, S. Rinderle-Ma, P. Dadam, X. Zhou, S. Pearson, A. Ferscha, S. Bergamaschi, and I. F. Cruz, editors, *On the Move to Meaningful Internet Systems: OTM 2012*, pages 305–322, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[8] J. Gama, P. P. Rodrigues, E. Spinosa, and A. Carvalho. Knowledge Discovery from Data Streams. *Web Intelligence and Security - Advances in Data and Text Mining Techniques for Detecting and Preventing Terrorist Activities on the Web*, pages 125–138, 2010.

[9] P. Harremoës and G. Tusnády. Information divergence is more χ^2-distributed than the χ^2-statistics. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 533–537, July 2012.

[10] B. Hompes, J. Buijs, W. Van der Aalst, P. Dixit, and J. Buurman. Discovering deviating cases and process variants using trace clustering. In *Proceedings of the 27th Benelux Conference on Artificial Intelligence (BNAIC), November*, pages 5–6, 2015.

[11] M. Jans, J. M. van der Werf, N. Lybaert, and K. Vanhoof. A business process mining application for internal transaction fraud mitigation. *Expert Systems with Applications*, 38(10):13351 – 13359, 2011.

[12] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132 – 156, 2017.

[13] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst. *Process and Deviation Exploration with Inductive Visual Miner*, volume 1295 of *CEUR Workshop Proceedings*, pages 46–50. CEUR-WS.org, 2014.

[14] A. Maaradji, M. Dumas, M. Rosa, and A. Ostovar. Fast and accurate business process drift detection. In *Proceedings of the 13th International Conference on Business Process Management - Volume 9253*, pages 406–422, Berlin, Heidelberg, 2015. Springer-Verlag.

[15] A. Ostovar, A. Maaradji, M. La Rosa, A. H. M. ter Hofstede, and B. F. V. van Dongen. Detecting drift from event streams of unpredictable business processes. In I. Comyn-Wattiau, K. Tanaka, I.-Y. Song, S. Yamamoto, and M. Saeki, editors, *Conceptual Modeling*, pages 330–346, Cham, 2016. Springer International Publishing.

[16] E. Rojas, J. Munoz-Gama, M. Sepúlveda, and D. Capurro. Process mining in healthcare: A literature review. *Journal of Biomedical Informatics*, 61:224 – 236, 2016.

[17] M. L. Rosa, H. A. Reijers, W. M. van der Aalst, R. M. Dijkman, J. Mendling, M. Dumas, and L. García-Bañuelos. Apromore: An advanced process model repository. *Expert Systems with Applications*, 38(6):7029 – 7040, 2011.

[18] A. Rozinat and W. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64 – 95, 2008.

[19] A. Seeliger, T. Nolle, and M. Mühlhäuser. Detecting concept drift in processes using graph metrics on process graphs. In *Proceedings of the 9th Conference on Subject-oriented Business Process Management*, S-BPM ONE '17, pages 6:1–6:10, New York, NY, USA, 2017. ACM.

[20] G. M. Tavares, V. G. T. da Costa, P. Ceravolo, V. E. Martins, and S. Barbon Jr. Anomaly detection in busi-ness process based on data stream mining. In *In Proceedings of XIV Brazilian Symposium on Information Systems (SBSI'18)*, Caxias do Sul, RS, Brazil, 2018. SBC.

[21] W. van der Aalst, A. Adriansyah, and B. van Dongen. Causal nets: A modeling language tailored towards process discovery. In J.-P. Katoen and B. König, editors, *CONCUR 2011 – Concurrency Theory*, pages 28–42, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[22] W. van der Aalst, M. Schonenberg, and M. Song. Time prediction based on process mining. *Information Systems*, 36(2):450 – 475, 2011. Special Issue: Semantic Integration of Data, Multimedia, and Services.

[23] W. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, Sept 2004.

[24] W. e. a. van der Aalst. Process mining manifesto. In *Business Process Management Workshops*, pages 169–194, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[25] W. M. P. van der Aalst. *Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management*, pages 1–65. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[26] W. M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer Publishing Company, Incorporated, 1st edition, 2011.

[27] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst. The prom framework: A new era in process mining tool support. In G. Ciardo and P. Darondeau, editors, *Applications and Theory of Petri Nets 2005*, pages 444–454, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[28] 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.*, 10(2):151–162, Apr. 2003.