Submission date: 25/11/2019
Resubmission date: 30/04/2020
Resubmission date: 29/06/2020
Camera ready submission date: 05/07/2020

1st round notification: 09/03/2020
2nd round notification: 09/06/2020
3rd round notification: 01/07/2020
Available online: 06/07/2020
Publication date: 31/07/2020

Section: extended version
Extended versions of papers presented at the XV Brazilian Symposium on Information Systems (SBSI 2019)

Comparing Concept Drift Detection with Process Mining Software

Nicolas Jashchenko Omori[1], Gabriel Marques Tavares[2], Paolo Ceravolo[2], Sylvio Barbon Jr.[1]

[1]Computer Science Department – State University of Londrina (UEL)
Londrina, Paraná – Brazil

[2]Dipartimento di Informatica – Università degli Studi di Milano (UNIMI)
Milan, Italy

{omori, barbon}@uel.br, {gabriel.tavares, paolo.ceravolo}@unimi.it

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 and software 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, comparing their differences, advantages, and disadvantages by testing against a log taken from a Process Control System. Thus, by highlighting the trade-off between the software, the paper gives the stakeholders the best options regarding their case use.*

Keywords. *Business Process Management, Process Mining, Concept Drift*

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, it may impose new challenges [Krawczyk et al. 2017]. 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 [Gama et al. 2010].

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) [van der Aalst 2011]. In traditional PM techniques, one has access to data from event logs recorded by systems controlling the execution of processes [Tavares et al. 2018]. Thus,

Cite as: Omori, Nicolas Jashchenko., Tavares, Gabriel Marques., Ceravolo, Paolo., & Barbon, Sylvio (2020). Comparing Concept Drift Detection with Process Mining Software. iSys: Revista Brasileira de Sistemas de Informação (Brazilian Journal of Information Systems), 13(4), 101-125. DOI: 10.5753/isys.2020.832

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 [Seeliger et al. 2017]. Identifying a drift, i.e. 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 [Maaradji et al. 2015].

In recent years the implementation of CD techniques for PM has received attention. Particularly, two algorithms [Bose 2012, Ostovar et al. 2016] for concept drift detection are widely available on popular PM software.

The first method was proposed by Bose et al. [Bose 2012] 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. [Ostovar et al. 2016] 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.

Furthermore, there are other process drift detection methods proposed in state-of-the-art [Accorsi and Stocker 2012, Maaradji et al. 2015, Zheng et al. 2017, Seeliger et al. 2017, Liu et al. 2018, Barbon Junior et al. 2018, Tavares et al. 2019]. As these methods are not embedded in PM software, it is noticeable that they usually do not have a user interface, which decreases their spread among non-expert users. To enhance the comparison proposed in this paper, we selected [Zheng et al. 2017] and [Tavares et al. 2019] since they are available as open-source and can be used in this work.

Zheng et al. [Zheng et al. 2017] proposed a three-stage approach named as Tsinghua Process Concept Drift Detection (**TPCDD**). For this, each trace (i.e. a possible instance of the process execution) is represented by multiple relations, which are then partitioned. After this, all change points from the relations are combined to get the final drift result.

Tavares et al. [Tavares et al. 2019] presented an online technique for detecting drifts. The Concept Drift in Event Stream Framework (**CDESF**) joins different PM tasks in an online environment. For that, trace distances are calculated by comparing them to a global model that represents the current state of the process. Hence, these distances are fed to a density-based clustering algorithm that distributes the instances in the feature space. Finally, the discovery of new clusters represents the detection of new concepts in the stream, i.e., concept drift.

The chosen techniques for this study have a common 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 [Hompes et al. 2015]. In other words, the demand for a PM tool capable of supporting accurate CD analysis with a straightforward setup process and of-

fering human-friendly interpretations is increasing.

This work is an extension of Omori et al. [Omori et al. 2019]. The original work aims at comparing two highly used methods for concept drift detection in business processes, ProM's **Concept Drift** plugin [Bose 2012] and **ProDrift** [Ostovar et al. 2016]. For that, the comparison follows several perspectives, such as performance, sensitivity to drifts, hyperparameters impact, and software interface. The main characteristic of the original approaches is that both have an easy-access interface, making them more suitable for non-expert users.

The improvement over the original work proposes the addition of two upgrades: the Process Control System event log was extended with one year worth of recordings, and two more approaches were added to the comparison, TPCDD [Zheng et al. 2017] and CDESF [Tavares et al. 2019]. The new approaches follow more academic standards, which might leverage concept drift detection at the cost of not offering as much support for non-expert users. Therefore, this work presents a more in-depth evaluation of the approaches, considering a trade-off between performance and user interaction. Finally, with the addition of more events in the dataset, more scenarios can be explored by the algorithms.

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 compares them over the dataset experiment. Lastly, Section 5 concludes the paper, emphasising open avenues for future work.

2. Process Mining

In the 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, which focus on data analysis [van der Aalst 2011]. PM main focus is to discover, monitor and enhance business processes towards a clear understanding of the process [van der Aalst 2004].

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 [Buijs et al. 2012]. 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 [van der Aalst et al. 2004], the inductive miner [Leemans et al. 2014], the heuristic miner [Weijters and van der Aalst 2003], among oth-

ers) for model extraction and the final model may be different within different discovery techniques.

Monitoring processes is commonly referred to as conformance checking, which aims at detecting inconsistencies amongst a process model and an event log corresponding to the same process [Rozinat and van der Aalst 2008]. Moreover, conformance may provide the 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 [Aalst, van der et al. 2012]. An enhancement technique may either repair the model by updating its relations or extend the model by adding new information, such as timestamps [van der Aalst 2011].

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* [van der Aalst 2011]). 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* [van der Aalst et al. 2011b]). Let $\mathcal{C}$ be the *case universe*, that is, the set of all possible case identifiers. An *event log* is a set of cases $L \subseteq \mathcal{C}$ 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* [van der Aalst et al. 2011a]). 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 Challenges in Process Mining

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 [Barbon Junior et al. 2018].

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 [Gama et al. 2010]. 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. 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* [Krawczyk et al. 2017]). 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 distribu-

tions [Krawczyk et al. 2017, Gama et al. 2010]. 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 [van der Aalst 2012]. The utmost consequence of drift is an outdated model, incapacitating its capability of recognising the new behaviour.

Maaradji et al. [Maaradji et al. 2015] 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. [Seeliger et al. 2017], 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* [Krawczyk et al. 2017, Gama et al. 2010]). 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. [Bose et al. 2014] 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. This distinction is important since online methods provide a real-time response to process behaviour. For instance, anomalous executions can be detected as they happen, and new concepts can be learned on the fly. This way, online process mining techniques provide better support for the decision-making task in organisations, further saving resources and time.

2.2. Concept Drift Detection in Process Mining

In the following subsections, we present and explain each of the different methods used for Concept Drift detection in this comparison.

2.2.1. ProM's Concept Drift plugin

Bose et al. [Bose 2012] method, available in ProM's Concept Drift plugin, is grounded on the premise that the relationship between activities can characterise event logs. For that, the dependencies are used in the feature extraction and selection part of the framework. These 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*. This way, event log features can be drawn from these relationships. There are four types of features proposed: Relation Type Count, Relation Entropy, Window Count and J-measure. The first two are global, while the latter two are 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, overlapping or not, and being continuous or non-continuous

(there can be a gap between populations). 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.

It is relevant to notice that the technique performs an offline analysis in the event log. This happens because it depends on the complete log to form the traces for feature extraction.

2.2.2. Apromore ProDrift plugin

Ostovar et al. [Ostovar et al. 2016] implementation, in Apromore, assumes 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. Based on this assumption, they propose to represent the process behaviour using the $\alpha+$ relations [Alves De Medeiros et al. 2004], which are a set of rules to represent different relations between activities.

Thus, the technique uses two adaptive-sized sliding windows to create sub-logs, 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 [Harremoës and Tusnády 2012] 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. This approach consumes a stream of events, meaning that it performs online drift detection.

2.3. Concept Drift Event Stream Framework

Tavares et al. [Tavares et al. 2019] proposed an online framework to deal with concept drifts, anomaly detection and process monitoring. The approach relies on a graph-based representation of the business process, which is updated with new events from the stream. The technique has the following steps: first, the trace is compared with the process model to retrieve their distances. These distances have two perspectives, time and trace, that are separately calculated by creating histograms with, respectively, the time difference between a pair of activities to the mean time difference of that pair and the relative frequency of each pair of activities from the trace to the relative frequency of the pair in the process model. Then, these distances values are fed to a density-based stream clustering algorithm, DenStream [Cao et al. 2006]. DenStream represents common and anomalous behaviour based on the clusters densities. Finally, either the detection of new clusters or fading of existing clusters represent concept drift. Following a time window, events are released from memory, and the graph model is updated.

2.4. Tsinghua Process Concept Drift Detection

Zheng et al. [Zheng et al. 2017] technique considers that an event log can be modelled using two types of order relations based on traces: Direct Success Relations and Weak

Order Relations. The former indicates that an activity directly succeeds another activity, while the latter indicates that an activity comes before another, but not necessarily directly before. The technique transforms the event log into a set of order relations between each pair of activities, named a relation matrix. Then it analyses the variations and trends for each of the relations on the matrix and chooses candidate drift points by calculating the frequency of a relation (i.e. if it always, sometimes or never happens) over a specified time window. These candidate drift points are then used as input for DBSCAN, a density-based clustering algorithm, that considers the densest clusters as being filled with real drift points. This technique also performs an offline analysis as it depends on complete traces for processing.

3. Materials and Methods

The following subsections give an overview of the materials used in this research, such as the event log used for the testing, a review of the available process mining software and how the comparison is defined. This is empirical research with the goal of comparing the different available techniques for detecting concept drifts in a business process. The main limitation of the study lies in the use of a single, not labeled event log, meaning that we do not know where and how many drifts are in the event log, so it is not possible to check for the accuracy of each technique.

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 [Braha 2013]. Process Control System often exhibits regular and predictable events, but some dynamic changes for improving the quality of services delivered are required. In this scenario, dealing with CD in Process Control System demands 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 of 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 algorithm with standard parameters. The extracted event log contains process instances from January 2014 through August 2019, and the Causal net represents all the process behaviour in this period.

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

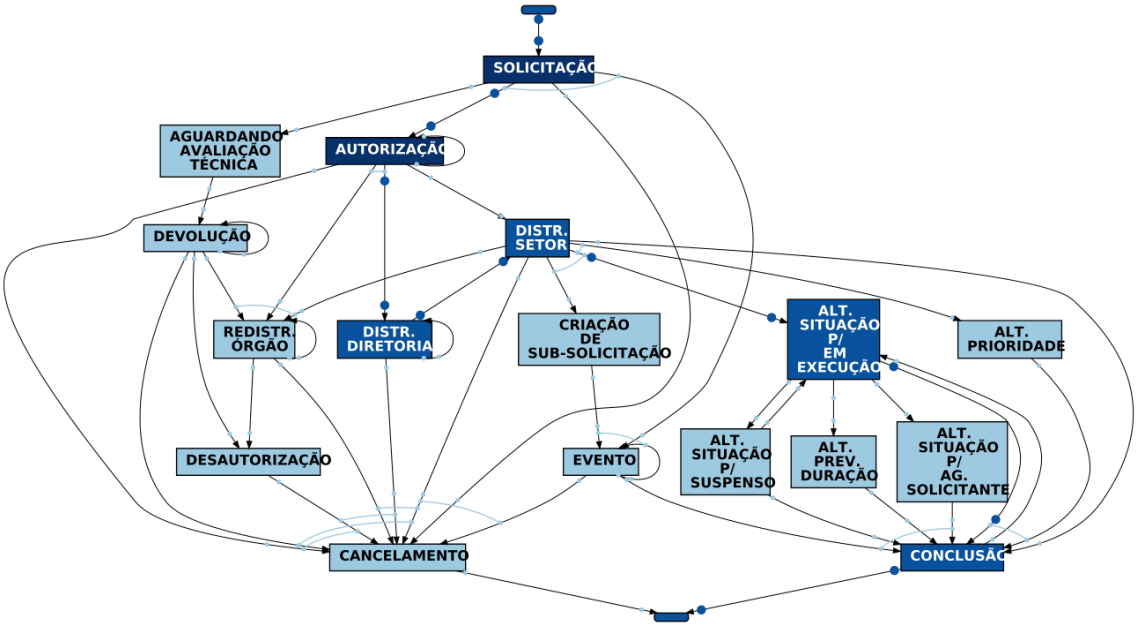

Figure 1. Causal net that represents the event log

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. In some particular cases, a process that is concluded can be reactivated by the requester if the result was deemed inadequate, so the process comes back to a stage of execution, marked by the activity ALT. SITUAÇÃO P/ EM EXECUÇÃO.

Table 2 shows the number of appearances of each activity through the log. The most evident pattern is a group of activities occurring over sixteen 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 unusual activities are shown in Table 2's tail, representing rare cases where the process flow is odd.

As an extension of the previous analysis, we have selected the three most common traces in the event log. The trace ⟨SOLICITAÇÃO, AUTORIZAÇÃO, DISTR. DIRETORIA, DISTR. SETOR, ALT. SITUAÇÃO P/ EM EXECUÇÃO, CONCLUSÃO⟩ occurs 14085 times, representing the regular execution flow of answered requests. The second most occurring one is ⟨SOLICITAÇÃO, AUTORIZAÇÃO, DISTR. SETOR, ALT. SITUAÇÃO P/ EM EXECUÇAO, CONCLUSÃO⟩, which occurs 331, is a slight variation in the most frequent trace, without the activity DISTR. DIRETORIA. And the third most ocurring, ocurring 294 times, is ⟨SOLICITAÇÃO, AUTORIZAÇÃO, DISTR. DIRETORIA, DISTR. SETOR, CANCELAMENTO ⟩ representing cancelled

Table 2. Occurrence count in DICT event log from January 2014 to August 2019

Activity	Number of Occurrences
AUTORIZAÇÃO	17100
DISTR. SETOR	16964
DISTR. DIRETORIA	16901
SOLICITAÇÃO	16844
ALT. SITUAÇÃO P/ EM EXECUÇÃO	16046
CONCLUSÃO	16037
CANCELAMENTO	808
EVENTO	588
DEVOLUÇÃO	543
DESAUTORIZAÇÃO	524
REDISTR. ÓRGÃO	445
CRIAÇÃO DE SUB-SOLICITAÇÃO	114
ALT. PREV. DURAÇÃO	86
ALT. SITUAÇÃO P/ SUSPENSO	63
ALT. PRIORIDADE	53
ALT. SITUAÇÃO P/ AG. SOLICITANTE	34
ALT. SITUAÇÃO P/ AG. SUBSOLIC	8
ALT. SITUAÇÃO P/ AG. SERV. EXT	7
ALT. SITUAÇÃO P/ AG. MATERIAL	4
AGUARDANDO AVALIAÇÃO TÉCNICA	4

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.

The frequency of the transitions between a pair of activities can also provide valuable information about the process. The most common transitions are the ones that happen in the two most occurring traces, representing the regular behaviour and happening over 15000 times each, the exception being AUTORIZAÇÃO–DISTR. SETOR, happening 405 times. Next, there are pairs DESAUTORIZAÇÃO–AUTORIZAÇÃO and DISTR. SETOR–CANCELAMENTO, the former representing an evaluation about an execution that was not authorized and the latter representing the cancelled traces, commonly happening after an execution was distributed to a sector of the organisation. The pair of transitions occurs 378 and 346 times, respectively.

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 (30.34) 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 four weeks, which is reasonable in a university environment with different inner

organisations involved in the business process.

Table 3. Statistics from DICT event log

Statistics	Value
Total cases	16840
Mean cases per day	30.34
Max cases per day	157
Total events	206346
Mean events per day	100.95
Max events per day	587
Mean trace length	6
Max trace length	24
Mean case duration (in days)	28.88

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. [van Dongen et al. 2005] proposed ProM[1], an open-source, extensible, process mining framework, to provide a set of techniques for exploring event logs. ProM is widely used in the process mining community, as evidenced by several publications [Jans et al. 2011, Rojas et al. 2016], and the 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.

One of these plugins is the Concept Drift plugin, used for detecting and identifying points of Concept Drift in a business process by generating populations from the process and applying statistical tests to compare two populations at a time and, if the populations are significantly different, it is considered as a drift point.

Also inspired by the need of PM framework, Apromore[2] was launched firstly as an advanced process model repository [Rosa et al. 2011] 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 [Maaradji et al. 2015, Ostovar et al. 2016],

[1]http://www.promtools.org/
[2]http://apromore.org/

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[3].

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, that is, performing an offline drift detection 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 are other softwares that employ process mining techniques, such as Disco[4], Minit[5] and QPR[6], 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.

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, i.e. how many events are selected for a population. We have used 58, 263, 1135 and 2270 events for the size. 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;

[3]https://github.com/apromore/ApromoreCode
[4]https://fluxicon.com/disco/
[5]https://www.minit.io/
[6]https://www.qpr.com/solutions/process-mining

- 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 which 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;
- 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. There are options of pairs that are invalid, and all valid options were used.

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 with the time period option.

On CDESF, many parameters modify the behaviour of the stream and drift detection. The parameters are explained in the documentation provided with the tool. However, there are no suggested values for each parameter, with the default values plugged directly in the code. On the tests, the following values were used:

- TH: sets a time horizon window (in seconds) that controls the ocurrence of the checkpoints. The values 43200, 86400, 432000 and 1296000, respectively 3, 6, 30 and 90 days, were used;
- Epsilon: sets the maximum distance in which an instance is part of a cluster. The values 0.05, 0.1 and 0.15 were used;
- Lambda: determines the importance of the historical data for the current clusters. The values 0.1, 0.15 and 0.2 were used;
- Beta: threshold that, based on the weight of a micro-cluster, defines if it is an outlier or potential micro-cluster. The values 0.2, 0.3 and 0.4 were used;
- Mu: is the minimum weight for a neighborhood to be considered a core object. The values 2, 3 and 4 were used;
- Stream Speed: is the number of objects that are considered per time unit. The values 500, 1000 and 1500 were used;

Except for TH, the other parameters regulate the DenStream clustering algorithm. As in other tests, a grid methodology was used, resulting in 972 different configurations.

For TPCDD's approach, the software provides only two parameters that can be changed in the execution, and there are no suggestions on how these parameters should be set beyond the default values. Therefore, the grid methodology was used for testing and the available parameters, and the values used on the tests were:

- Minimum Window Size: Sets the minimum amount of traces needed for the detection of the features. The default value is 100 and the tests were run using 10, 25, 50, 75, 100, 150, 200, 250 and 300 for the minimum size;
- DBSCAN Radius: It is a parameter that sets the *eps* parameter for the DBSCAN algorithm. The *eps* sets the maximum radius of the DBSCAN cluster. The default value is 10 and the values 3, 5, 8, 10, 15, 20, 25 and 30 were used for the tests.

As there are only two parameters, more different values could be used on the tests for each option, resulting in 72 different tests.

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 a 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 and unique drifts are tracked. 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 8680 drift points were found, being 1209 of them unique. Considering that the event log contains 206346 events and 16840 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, 27 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 17581, which happened on December 3, 2014. To verify this drift's veracity, we divided the event log before and after said point, starting and ending at the preceding and succeeding drift points, 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 2 and 3 show the Causal nets before and after the found drift point. It is clear that the Causal net from Figure 3 is more complex than the one in Figure 2, presenting more arcs and connections within activities. One example is that both `CANCELAMENTO` and `DEVOLUÇÃO` activities are not present in the first Causal net but are present 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, Figures 3 and 2 show a drift point since the behaviour observed before and after the point is different, meaning the process has changed during its execution.

Figure 2. Process model before drift point on December 3

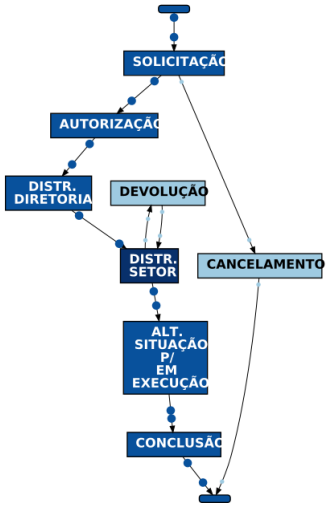

Figure 3. `CANCELAMENTO`, `DEVOLUÇÃO` did not occur before the drift point

As of ProM's Concept Drift plugin, we used the same testing method and, for the 109 tested configurations, 1860 drift points were found, a mean of 17.22 drift points per configuration. Of the total 1860 points, 413 points are unique (22% of the total), indicating consistency in the points that are detected as drifts. As for the most common

drift point, happening in February 19, 2015, had a subtler difference in models compared to ProDrift's result. In the Figures 4 and 5 the difference is found in sequences of activities that are present in the process before the drift point and are not in the process after the drift point, as well as the activity `ALT. SITUAÇÃO P/ SUSPENSO` not happening after the drift point. It can also be noted that ProM's most common drift point occurs in a similar period to ProDrift's, nearing the end of the year 2014 and the start of 2015, indicating a real drift in the process.

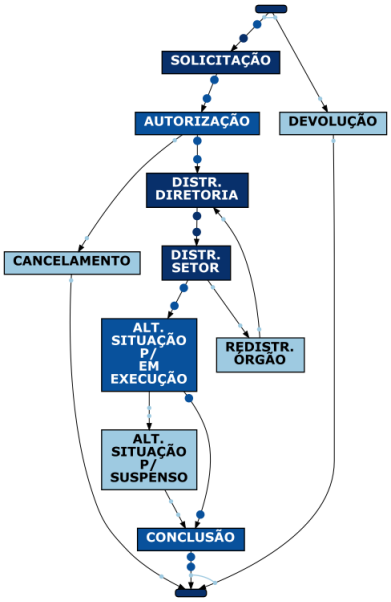

Figure 4. Process model before drift point on February 19

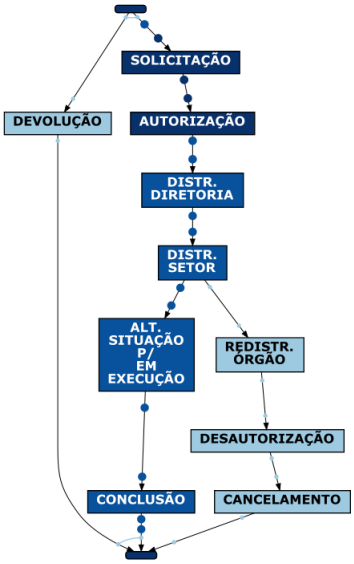

Figure 5. The model is changed to a simpler version

CDESF was, by far, the technique that had the largest number of different combinations for the available parameters, with 972 tested configurations that detected, in total, 9594 drifts, resulting in a mean 9.87 drifts per configuration. The configuration that detected the largest amount of drift points detected 85 drifts, a number that is lower than all of the other techniques. There were 301 configurations in which there were no drifts found, 30% of the tested configurations. The amount of unique drift points is 1183, 12% of the total number of drifts, meaning that the technique, most of the time, finds the same drift points, leaving an easier analysis for the decision-maker. The most common drift point occurs at February 19, 2014, this date being exactly same the time of ProM's most common drift, and being in the same time period of ProDrift's most common drift. This further indicates the occurrence of a real drift at the time, and the Figures 6 and 7 shows the drift point as it was found by CDESF.

TPCDD's approach yields a higher quantity of detected drift points, with 11098 drifts across 73 parameter configurations, resulting in a mean of 154.13 drift points. The maximum amount of drifts in a single configuration is 987 drifts. Considering the time between the first and the last event recorded on the log, 2044 days, the technique detects a drift every 2 days in average. Hence, the method sensitivity is very high, something further reinforced by the fact that there were no test cases in which no drifts were found.

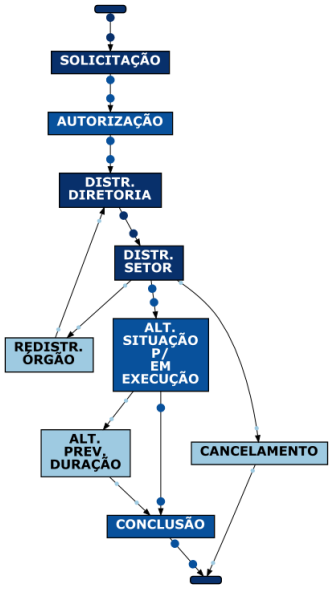

Figure 6. Process model before drift point on February 19

Figure 7. Difference in arcs and `ALT. SITUAÇÃO P/ SUSPENSO` did not happen after the point

There are 3960 unique drifts, around 35% of the total, but this number could be inflated because of the high sensitivity and the variations in the Minimum Window Size parameter, causing slight offsets in the exact location of the drift point. The most common drift point for this technique happened in October 22, 2018 and, as its shown in the Figures 8 and 9, there is a big difference in the complexity of the process after the drift point.

The results are summarized in the Table 4, indicating the number of configurations, the mean number of found drifts, the minimum and maximum in a single configuration and the number of unique drifts.

Table 4. Statistics from DICT event log

	ProDrift	ProM	CDESF	TPCDD
Number of Tested Configurations	160	109	972	73
Mean Drift Points	54.25	17.22	9.87	154.3
Maximum Drift Points	335	167	85	987
Minimum Drift Points	0	0	0	26
Unique Drift Points	1209	413	1183	3960

4.2. Hyperparameters dependence

Table 5 presents the hyperparameters related to each tool used. By comparing the 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, this contrasts with methods with fewer parameters, which are more user-friendly and easier for non-experts to use.

Table 5. Comparison Summary

Characteristic	ProM	ProDrift	CDESF	TPCDD
Detection Method	Kolmogorov-Smirnov Test, Mann-Whittney Test or t-Test	G-Test	DenStream clustering	DBSCAN clustering
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	Adaptive Window, parameter: Time Horizon	Fixed Size, parameter: Minimum 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	Graph distances, parameters: none	Direct Success and Weak Order Relations, parameters: none
Sensitivity	p-value, parameters: p-value threshold	*Very Low*, *Low*, *Medium*, *High* and *Very High*, parameters: Cumulative change (%), Drift detection sensitivity	Cluster weight, parameter: Mu	DBSCAN Radius
Noise	-	Filtering, parameters: Relation noise filter (%)	Cluster-based anomaly detection, parameter: Beta	-
Drift Characterization	Sudden Drifts and Gradual Drifts	All drift points, parameter: Drift Characterization (on or off)	-	-
Stream Type	Trace stream (Offline)	Both streams of events and traces can be used. Parameters: Trace-based (Runs) or Event-based ($\alpha+$) (Online)	Event stream (Online)	Trace stream (Offline)

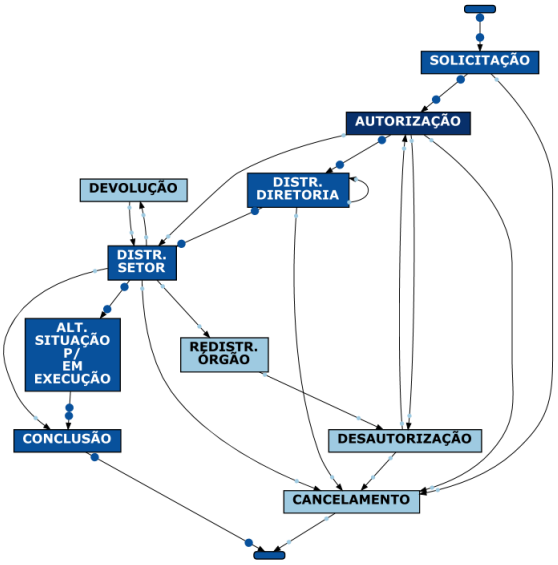

Figure 8. Process model before drift point on October 22

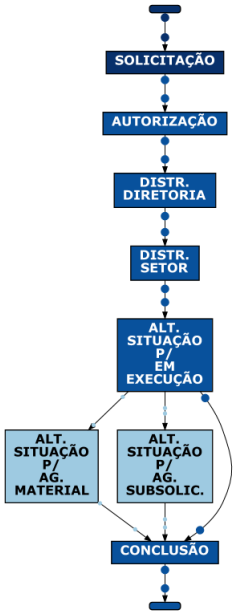

Figure 9. Difference complexity is notable after the drift point

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 1.22 drift points per configuration while tests with Mann-Whittney drift detection algorithm average 33.22 drifts per configuration. Other hyperparameters, such as choosing between Window Count or J-Measure, did not affect the number of drifts but affected the location of such drifts slightly.

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. Larger 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 manoeuvre. 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.

On CDESF, the Epsilon and the Stream Speed were the most influential hyperparameters, while the TH and the Beta hyperparameters had less effect on the number of drifts detected. Epsilon sets the radius of the clustering algorithm, with a higher value detecting fewer drifts, as they are considered part of the same cluster. Stream Speed works similarly to a selection of window size, with smaller values detecting more drifts. Even if the TH and Beta were the least influential hyperparameters, they still affected drift results,

indicating that every single parameter matters regarding the sensitivity.

As for the TPCDD, in a similar fashion to ProDrift, the window size is a very influential hyperparameter regarding the sensitivity to drifts, since a small window makes the algorithm to consider any small deviation as a drift point. In the same way, the DBSCAN radius parameter is also very influential on the sensitivity, affecting directly the *threshold* of the drift detection, as with a big radius more traces are grouped in a single cluster, shrinking the number of clusters and detecting fewer drifts, with the opposite phenomenon happening when a small radius is set.

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 10. 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.

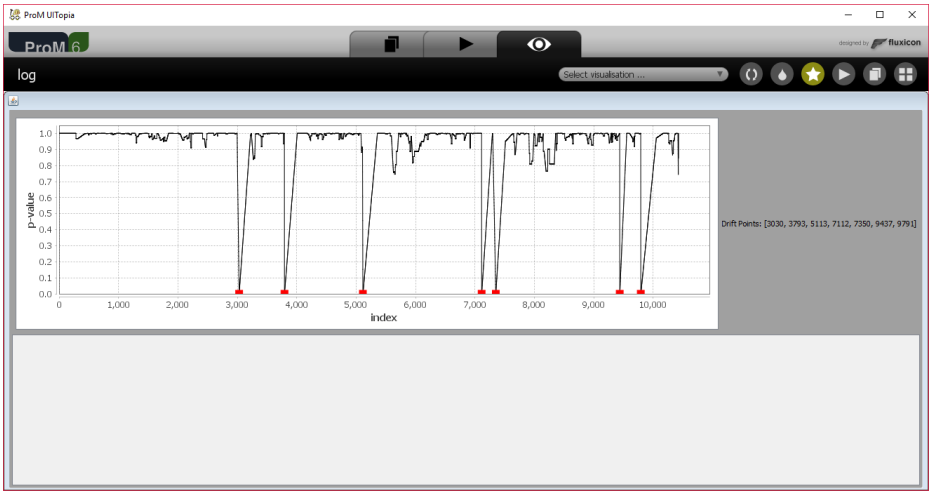

Figure 10. ProM's Concept Drift graphical interface

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[7]. To use the ProDrift plugin, the event log has to be imported to the server in the following formats: *CSV*, *XML*, *MXML* or *XES*. There is only

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

one page of configurations to be chosen, making the software more 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 11. Despite having a command-line version, it is still difficult to automate the tests, as every test outputs to the system's standard output and a window have to be manually closed every time. That being said, both tools provide a accessible and easy use of the graphical interface.

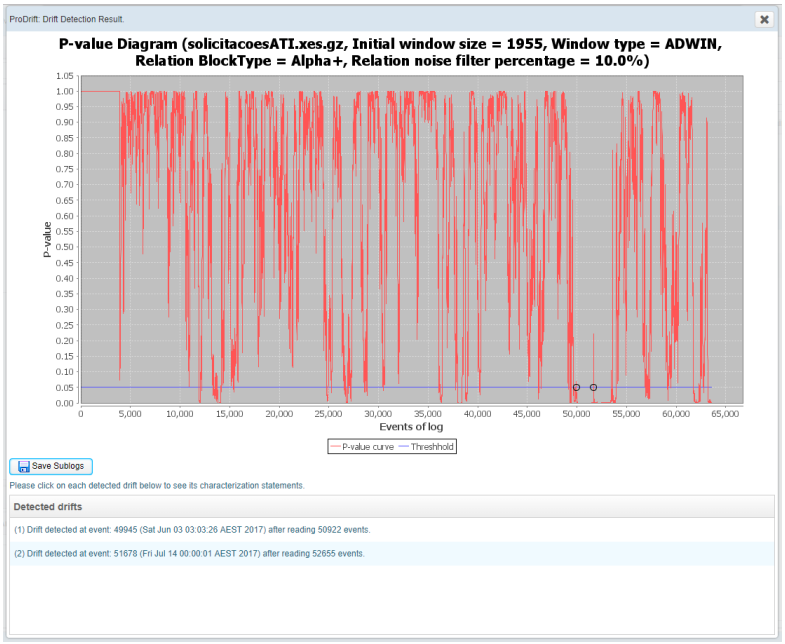

Figure 11. ProDrift output graphical interface

As for CDESF, the software must be downloaded from the source repository[8] and run from the command line. The code is written in Python, so it can be run in any platform, given that Python itself and the necessary libraries are installed. Alongside the code, there is a file containing the documentation with information about the functions as well as their dependencies. As the input, the event log must be in a specific *CSV* format, specified in the main file of the program. The usability is limited for non-expert users, as one need to open the source code to change the default parameters. The output is saved within folders named after the used parameters, and these folders contain the plots of the models generated for each checkpoint. The identification of drift points is not straightforward, as there is only a file that must be run after the main program is finished. However, if the drift points are available, it is easy to check the difference by just comparing the plots of the checkpoints. Although it is not simple to run the program, it is easy to modify it for automatic testing with varying parameters.

Similarly to CDESF, TPCDD must also be downloaded from the source repository[9] and run from the command line. It is multi-platform, given that Python and its dependencies are installed on the system. In the guideline, there is a brief explanation of

[8]https://github.com/gbrltv/CDESF2
[9]https://github.com/THUBPM/process-drift-detection

how to set the two parameters and their default values. If there are any drift points, they are printed to the standard output in the form of a list. The output presents a list of indexes of the traces in the event log, in ascending time order. In some cases, these indexes could be useful, such as when the cases start in 0 and are incremented sequentially, but that is not always the case, so a list of *Case IDs* could be useful for further analysis in the drift points. It was by far the easiest to automate, as there are only two parameters, which can be set as arguments to the program, minimising manual work.

Regarding the ease of use, ProM and Prodrift provide a graphical user interface while CDESF and TPCDD must be run from the command line. It is important to note that ProDrift can also be run from the command line. While having a graphical interface can simplify the use and make the tool more accessible (apart from being included in a framework), the relative ease of testing multiple options automatically is a strong point for the command line tools. As there is a need for finding fitting parameters for the tools, there is a trade-off between the ease of use and the ease of testing different parameters. Moreover, academic tools are focused on researches who want to study the proposed technique while software solutions provide support for non-experts and stakeholders.

4.4. Threats to Validity

This work suffers from a set of threats to its validity. First, some of the techniques had as parameters numeric values without standardization on its ranges and, apart from the default values, there were little to no suggestions on documentation regarding the values for each parameter. Second, the real number of drift points in the Process Control System log is unknown, so a limitation of this work is the inability of comparing the precision of the drifts found by the techniques.

5. Conclusion

The amount of data that organisations have to handle nowadays is steadily growing. In addition to this growth, data is also continually changing, which means the 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 particular process by applying those tools in a real-life environment. For that, a Process Control System of a real organisation was used as an event log source. The findings show that there is a clear trade-off between usability, access and testing. Some tools provide better support for less experienced users but at the same time add several steps of configuration. Moreover, software tools do not necessarily deliver clear drift points, usually hiding timestamps. When selecting which tool to use for process drift detection, the user must take into account the solutions' outputs and the level of knowledge required for managing it, such as ProDrift and ProM are better choices for users who want an easy to use albeit not customizable tool, while CDESF and TPCDD are better choices if you have the know-how to be able to execute and customize it to your needs.

By summarizing the Concept Drift detection methods, this work provides a baseline for the proposal of new techniques that may solve the flaws shown on the methods,

such as delivering clear drift points to the end user.

6. Acknowledgements

This study was financed in part by Coordination for the National Council for Scientific and Technological Development (CNPq) of Brazil - Grant of Project 420562/2018-4 and Fundação Araucária (Paraná, Brazil).

References

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

Accorsi, R. and Stocker, T. (2012). Discovering workflow changes with time-based trace clustering. In Aberer, K., Damiani, E., and Dillon, T., editors, *Data-Driven Process Discovery and Analysis*, pages 154–168, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

Barbon Junior, S., Tavares, G. M., da Costa, V. G. T., Ceravolo, P., and Damiani, E. (2018). 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. International World Wide Web Conferences Steering Committee.

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

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

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

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

Cao, F., Estert, M., Qian, W., and Zhou, A. (2006). Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM.

Gama, J., Rodrigues, P. P., Spinosa, E., and Carvalho, A. (2010). 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.

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

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

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

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

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

Liu, N., Huang, J., and Cui, L. (2018). A framework for online process concept drift detection from event streams. In *2018 IEEE International Conference on Services Computing (SCC)*, pages 105–112.

Maaradji, A., Dumas, M., Rosa, M., and Ostovar, A. (2015). 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. Springer-Verlag.

Omori, N. J., Tavares, G. M., Ceravolo, P., and Barbon, Jr., S. (2019). Comparing concept drift detection with process mining tools. In *Proceedings of the XV Brazilian Symposium on Information Systems*, SBSI'19, pages 31:1–31:8, New York, NY, USA. ACM.

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

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

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

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

Seeliger, A., Nolle, T., and Mühlhäuser, M. (2017). 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. ACM.

Tavares, G. M., Ceravolo, P., Turrisi Da Costa, V. G., Damiani, E., and Barbon Junior, S. (2019). Overlapping analytic stages in online process mining. In *2019 IEEE International Conference on Services Computing (SCC)*, pages 167–175.

Tavares, G. M., da Costa, V. G. T., Ceravolo, P., Martins, V. E., and Barbon Jr, S. (2018). 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. SBC.

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

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

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

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

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

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

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

Weijters, A. J. M. M. and van der Aalst, W. M. P. (2003). Rediscovering workflow models from event-based data using little thumb. *Integr. Comput.-Aided Eng.*, 10(2):151–162.

Zheng, C., Wen, L., and Wang, J. (2017). Detecting process concept drifts from event logs. In Panetto, H., Debruyne, C., Gaaloul, W., Papazoglou, M., Paschke, A., Ardagna, C. A., and Meersman, R., editors, *On the Move to Meaningful Internet Systems. OTM 2017 Conferences*, pages 524–542, Cham. Springer International Publishing.