The Drift Analysis in the Process Mining with the Visual Drift Detection Tool

Anton Yeshchenko[1], Claudio Di Ciccio[1], Jan Mendling[1], Artem Polyvyanyy[2]
[1]Vienna University of Economics and Business, Vienna, Austria
{fanton.yeshchenko, claudio.di.ciccio, jan.mendlingg@wu.ac.at}
[2]The University of Melbourne, Parkville, VIC, 3010, Australia
{artem.polyvyanyy@unimelb.edu.au}

ABSTRACT: *The business process models change in the recent past to where the shift from concept drift into process mining occurs over a period of time. The studies conducted till now have not addressed the requirements and not yet addressed the challenges of drift categorization, drilling-down, and quantification. Throughout this work, we present a new software tool to analyze process drifts, called Visual Drift Detection (VDD), which fulfills these requirements. The tool is of benefit to the researchers and practitioners in the business intelligence and process analytics area, and can constitute a valuable aid to those who are involved in business process redesign endeavors.*

Keywords: Drift Detection, Process Mining, Time Series Analysis, Change Point Detection, Declarative Process Models

DOI:10.6025/ed/2020/9/1/24-28

Received: 28 August 2019, Revised 2 December 2019, Accepted 15 December 2019

1. Introduction

The availability of data has extended conceptual modeling as a research field of manually created models with automatic techniques for generating models from data. Process mining is one of these recent extensions that is concerned with providing transparency of how the businesses operate based on real-world event data. Process discovery is a branch of process mining that takes as input event logs, i.e., collections of event sequences (traces) wherein every event corresponds to an activity execution, and returns the model that best describes the process generating the event log. However, process mining analyzes aggregated snapshots of sequentially stored process executions. Therefore, it can overlook the behavioral changes that occur in the time lapse during which those data were gathered. In data mining, such a change over time is called a drift. A drift is a concept that process mining has addressed only to a limited extent so far.

Recent works such as that of Maaradji et al. [7] and Ostovar et al. [8] have focused on the identification of specific drift types, based on the tracking of behavioral relations over time through statistical tests. In this paper, we present a novel technique for process drift detection, called Visual Drift Detection (VDD), which extends existing techniques by not only finding drifts but also helping the user recognize their type. Furthermore, it facilitates assessment of drifts through visual interpretation [10]. Our technique is founded in the formal rigor of temporal logic of DECLARE constraints [1,4] and time series analysis [9]. Key strengths of our technique are clustering of declarative behavioral constraints that exhibit similar trends of change over time and automatic detection of drift points. These features allow us to detect and explain drifts that would otherwise sneak undetected by other

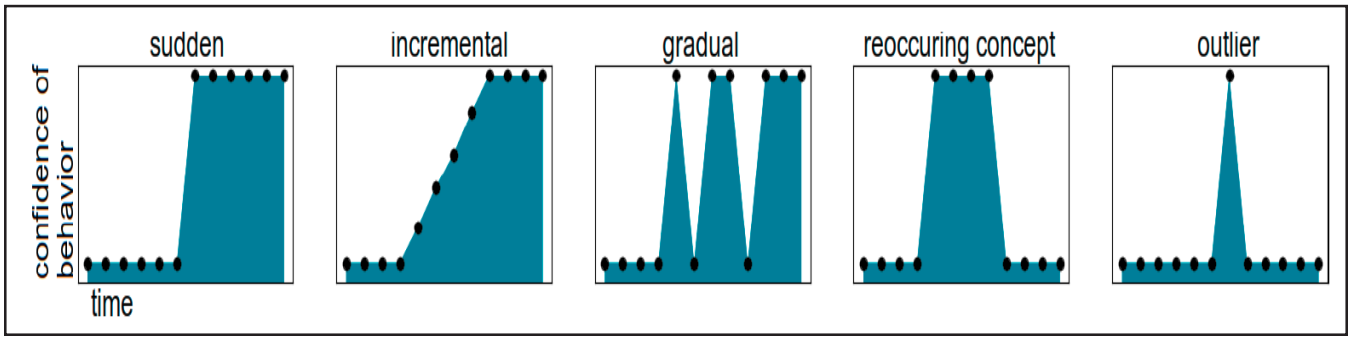

Figure 1. Drift types, cf. [5, Figure 2]; notice that an outlier is not a drift

techniques. In this paper, we outline our technique and illustrate the usage of the tool on a real-world data set publicly available on the 4TU Data Centre.[1]The event log contains events from a ticketing management process of the help desk of an Italian software company [8].We will henceforth refer to that data set as Italian help desk log.

This is a tool demonstration paper illustrating the software implementation of the VDD approach, which is detailed in [11] and shown in a dedicated video.[2]

2. Preliminaries

Process Drifts: A process drift is a notion in process mining for analyzing behavioral changes of business processes over time. The specific challenge is to not only spot a drift but also to classify it. Figure 1 shows established drift classes from data mining.Asudden drift is typically caused by an intervention, such as a new law exerting a change in the control flow. An incremental drift might result from a stepwise introduction of new routines. A gradual drift may yield from a new policy to be adopted in the enactment of the process. Finally, a reoccurring drift might result from specific measures taken at every occurrence of seasonal events, e.g., during holidays or festive days, weekends, night shifts, and so on. An outlier corresponds to an isolate episode and, as such, it does not qualify as a drift. Existing process mining techniques support these types of drifts only to a limited extent.

Declarative Process Constraints: In our approach, diagrams like those in Figure 1 are depicted considering the confidence level of process behavioral rules over time. In particular, we resort on the repertoire of rules provided by the declarative specification language DECLARE [1,4]. Examples of DECLARE constraints are RESPONSE(*a, b*), ALTERNATERESPONSE(*a, b*), and CHAINRESPONSE(*a, b*). The first constraint applies the RESPONSE template on tasks *a* and *b*, and states that if a occurs then b must occur later on within the same trace. In this case, *a* is named activation because it is mentioned in the "*if*" clause, thus triggering the constraint, whereas b is named target because it is in the "then" clause. RESPONSE(*a, b*) holds true in a trace like <*a, c, a, c, b*>. ALTERNATERESPONSE(*a, b*) asserts that RESPONSE(*a, b*) holds true and *a* does not recur before *b*, as in <*a, b, c, a, c, b*>. CHAINRESPONSE(*a, b*) imposes that RESPONSE(*a, b*) holds true and no other task occurs between *a* and *b*, as in <*a, b, c, a, b*>. Declarative process mining tools can measure to what degree constraints hold true in a given event log. One such measure is confidence [4]. It is computed as the number of activations that lead to a satisfaction of the constraint (e.g., the number of *a*'s eventually followed by an occurrence of *b* for RESPONSE(*a, b*) scaled by the percentage of traces in which the activation (*a*) occurs, to penalize constraints that are triggered sporadically. Confidence value ranges from 0 to 1.

3. The VDD Approach

Our multi-staged technique takes as input an event log (henceforth, log for short) and returns visual diagnostics on process

[1] https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb

[2] https://youtu.be/_AZpI_YTjO8

Cluster	Constraint	Activity 1	Activity 2	Min	Max	Mean
9	ChainPrecedence	Take in charge ticket	Create SW anomaly	0.0	100.0	42.8
	AlternatePrecedence	Assign seriousness	Create SW anomaly	0.0	100.0	49.0
11	ChainPrecedence	Take in charge ticket	Schedule intervention	0.0	100.0	9.9
	AlternatePrecedence	Assign seriousness	Schedule intervention	0.0	100.0	9.9
4	ChainResponse	Take in charge ticket	Wait	9.4	69.6	23.2
	NotSuccession	Resolve ticket	Wait	10.0	77.2	26.0
	NotSuccession	Wait	Assign seriousness	10.0	78.0	26.6
	NotSuccession	Wait	Take in charge ticket	9.8	73.3	22.1
	AlternateResponse	Assign seriousness	Wait	9.0	72.3	23.8
	AlternateResponse	Wait	Closed	8.3	61.4	22.5
	AlternateResponse	Wait	Resolve ticket	8.3	61.4	22.8
	AtMostOne	Wait		9.8	68.6	25.1

Table 1. Italian ticket log constraints; including min, max, and mean confidence

drifts. It consists of three main steps, which we shall explain through their application on the case study of the Italian help desk log.

In the first step, we sort the traces in the log by the timestamp of their respective first events. Thereupon, we extract a sub-log of a given window size from the first traces.We let the window slide over the log at a given step. From each sub-log we mine the set of DECLARE constraints and compute their confidence. In our case study, we set the window size to 100 and the sliding step to 50. For the sample log, we mine DECLARE constraints out of 90 sub-logs. For each sub-log, we compute the confidence of 2604 constraints, including those reported in Table 1.

In the second step, we extract multi-variate time series that represent the trends of the constraints' confidence. Then, we cluster those time series with hierarchical clustering [2] to find groups of constraints that exhibit similar confidence trends (henceforth, behavior clusters).We resort on the Pruned Exact Linear Time (PELT) algorithm [6] to detect change points in the whole multi-variate time series as well as within the behavior clusters. The change points denote process drifts. In our case study, we identify 16 clusters, including those in Table 1.We detect 3 process drifts for the overall multi-variate time series, and 12 within-cluster change points. We observe that the overall process drifts that our technique discovers include those that were found by ProDrift [8]. They occur approximately in the first half and towards the end of the time span.

In the third step, we plot graphical representations to visually identify and characterize the detected drifts.We create two categories of visual aids: Drift Maps and Drift Charts. Drift Maps display all drifts data on a two-dimensional plane. Figure 2(a) illustrates the Drift Map for overall drifts and Fig. 2(b) shows the drifts within behavior clusters. The x-axis is the time axis, while

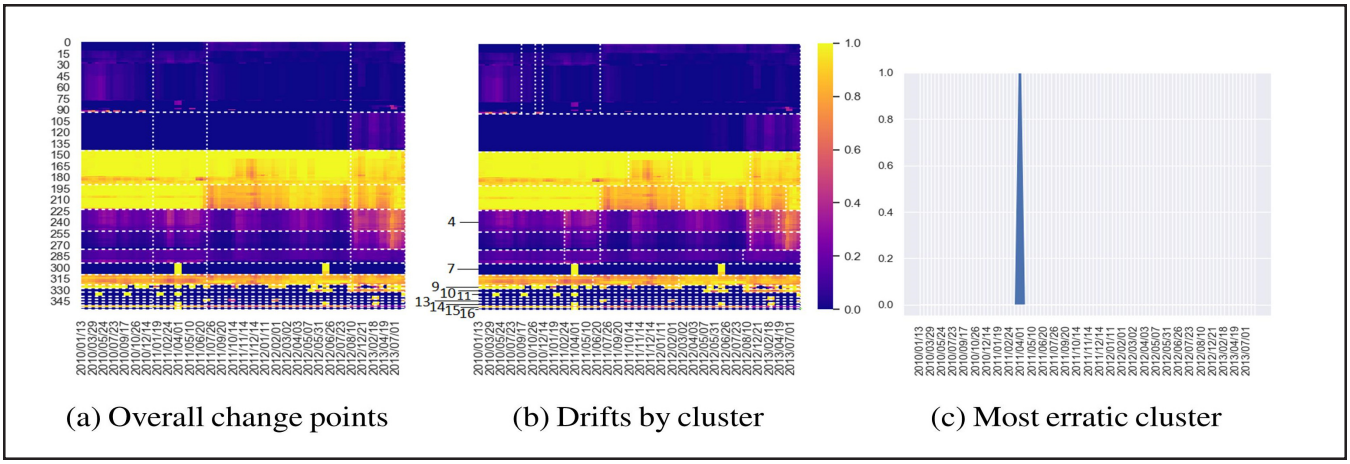

Figure 2. Italian help desk log VDD visualizations

every constraint corresponds to a point along the y-axis.We add vertical lines to mark the identified change points, i.e., drift points, and horizontal lines to demark clusters. Constraints are sorted by the similarity of the confidence trends. The values of the time series are represented through the plasma color-blind friendly color map [10], from blue (low peak) to yellow (high peak). Drift Charts (e.g., those in Fig. 3) have time on the x-axis and average confidence of the constraints in a cluster on the y-axis. We add vertical lines to denote change points as in Drift Maps. In order to find and pinpoint the most interesting (erratic) behavior clusters, we define a measure inspired by the idea of finding the length of a poly-line in a plot. The rationale is, straight lines denote a regular trend and have the shortest length, whilst more irregular, wavy curves evidence more behavior changes and their length is larger.

Detecting and explaining drifts: As illustrated by the VDD visualization in Figure 2(a), we detect a sudden change in the first quarter, in addition to the two identified by ProDrift. Following on that, we analyze the within-cluster changes (Figure 2(b)) and notice that the most erratic cluster contains an outlier, as shown by the spike in Figure 2(c).

In Figure 3, we illustrate the most erratic examples of behavior, and, in Table 1, we present the constraints that describe that specific behavior after applying the constraint minimization algorithm of [3]. Figure 3(a) shows an erratic behavior, which visually corresponds to the reoccurring concept classification from Fig. 1 (cluster 9). By examining the constraints that constitute this behavior, we can conclude that in the dates of the peak in Fig. 3(a) the activity Create SW anomaly always had Take in charge ticket executed immediately beforehand (CHAINPRECEDENCE). Also, we can conclude that before Create SW anomaly, the Assign seriousness activity was executed and no other Create SW anomaly occurred in between (ALTERNATEPRECEDENCE). Figure 3(b) (cluster 11) has four spikes, where Schedule intervention activities occurred. Immediately before Schedule intervention, Take in charge ticket occurred. Also, Assign seriousness had to occur before Schedule intervention recurred.We notice, however, that this cluster shows outlier behavior, due to its rare changes. Finally, Figure 3(c) (cluster 4) depicts a gradual drift until June 2012, and the incremental drift afterward.We notice that all constraints in the cluster have Wait either as an activation (e.g., with ALTERNATERESPONSE(Wait, closed) or as a target (e.g., with CHAINRESPONSE (Take in charge ticket, Wait).

4. Maturity, Documentation and Screencast

We implemented the VDD tool in Python[3], resorting on the scipy library for time-series clustering and the ruptures library for change point identification.We used the MINERful3 Java package for constraints discovery.We run our experiments using a laptop equipped with an Intel Core i5 at 2.40GHz × 4 with 16GB of RAM.With this modest hardware, the tool was able to process data and produce the analysis outcome in about 27 seconds using a real-size event log with 21438 events from 14 activities over 4580 traces. This indicates that the VDD tool has reached a fairly large degree of maturity as it performs well in terms of scalability. In future work, we will focus on the prediction of drifts in running processes and study how to improve the interpretability of depicted results.

We have created a project website for the VDD tool, from which it can be downloaded together with its sources.[4] It is free for academic and non-commercial use under the MIT license. On the project website, we provide documentation on its installation and first run. A screencast documenting its usage is available at https://youtu.be/_AZpI_YTjO8.

Acknowledgements

This work is partially funded by the EU H2020 program under MSCA-RISE agreement 645751 (RISE BPM). Artem Polyvyanyy was partly supported by the Australian Research Council Discovery Project DP180102839.

References

[1] van der Aalst, W.M.P., Pesic, M. (2006). DecSerFlow: Towards a truly declarative service flow language. In: *WS-FM. Lecture Notes in Computer Science*, volume 4184, p 1–23. Springer.

[3] https://github.com/cdc08x/MINERful

[4] https://github.com/yesanton/Process-Drift-Visualization-With-Declare

[2] Aghabozorgi, S., Seyed Shirkhorshidi, A., Ying Wah, T. (2015). *Time-series clustering - a decade review. IS* 53 (C) 16–38.

[3] Di Ciccio, C., Maggi, F.M., Montali, M., Mendling, J. (2017). Resolving inconsistencies and redundancies in declarative process models. IS 64, 425–446.

[4] Di Ciccio, C., Mecella, M. (2015). On the discovery of declarative control flows for artful processes. *ACM TMIS* 5 (4) 24, 1–24, 37.

[5] Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Comput. Surv*. 46 (4) 44, 1–44, 37.

[6] Killick, R., Fearnhead, P., Eckley, I.A. (2012). Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association,* 107 (500) 1590–1598.

[7] Maaradji, A., Dumas, M., La Rosa, M., Ostovar, A. (2017). Detecting sudden and gradual drifts in business processes from execution traces. *IEEE TKDE,* 29 (10) 2140–2154.

[8] Ostovar, A., Leemans, S.J., La Rosa, M. (2018). Robust drift characterization from event streams of business processes (2018) https://eprints.qut.edu.au/121158/

[9] Reinsel, G.C. (1993). Elements of multivariate time series analysis. Springer.

[10] Ware, C. (2012). Information visualization: perception for design. Elsevier.

[11] Yeshchenko, A., Di Ciccio, C., Mendling, J., Polyvyanyy, A. (2019). Comprehensive process drift detection with visual analytics. In: ER. Springer (2019), in print