VDD: A Visual Drift Detection System for Process Mining

Anton Yeshchenko, Jan Mendling
Vienna University of Economics and Business
Vienna, Austria
firstname.lastname@wu.ac.at

Claudio Di Ciccio
Sapienza University of Rome
Rome, Italy
claudio.diciccio@uniroma1.it

Artem Polyvyanyy
The University of Melbourne
Melbourne, Australia
artem.polyvyanyy@unimelb.edu.au

***Abstract*—Research on concept drift detection has inspired recent advancements of process mining and expanding the growing arsenal of process analysis tools. What has so far been missing in this new research stream are techniques that support comprehensive process drift analysis in terms of localizing, drilling-down, quantifying, and visualizing process drifts. In our research, we built on ideas from concept drift, process mining, and visualization research and present a novel web-based software tool to analyze process drifts, called Visual Drift Detection (VDD). Addressing the comprehensive analysis requirements, our tool is of benefit to researchers and practitioners in the business intelligence and process analytics area. It constitutes a valuable aid to those who are involved in business process redesign projects.**

I. Introduction

Process mining is a research field that is concerned with leveraging real-world event data for providing transparency of how business processes operate. 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 discovery analyzes event logs without distinguishing executions that are recent and that are far in the past. Therefore, it does not explicitly show the behavioral changes that occur in the time lapse during which those data is gathered.

These behavioral changes are a commonplace in the real-world scenarios and introduce additional challenges for the existing process mining techniques that are usually assume stable patterns of behaviour. If a drift is present in the data, it affects all stages of process mining namely discovery, conformance and enhancement [1] As a consequence, the *discovered models* are much more complex since they integrate behaviour that is present in different points in time. Using data affected by behavioral changes for process *conformance* also hinders the results by detecting non-compliant behaviour of the aggregated data from a process that might have been a norm for a particular time-span. The process *enhancement* using the event log containing changes would produce process models annotated with information that is not significant at all time stages. All these issues with process mining techniques could be alleviated or turned into the strengths by first analysing behaviour changes during process mining projects [2]. This is to the benefit of the process analyst who might quickly suffer from models being too complex inducing too high cognitive load to be comprehended in an accurate way [3].

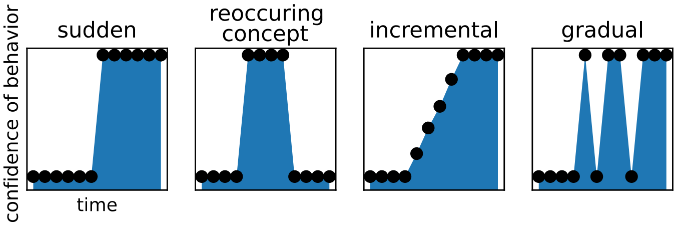


Figure 1: Drift types, cf. [7, Fig. 2]

Research on data mining has discussed changes over time and distinguishes different types of so-called *drift*. Drift analysis has been considered in prior research on process mining in the following way. Recent works include such contributions as Maaradji et al. [4] that use statistical tests in order to find sudden and gradual drifts, Zheng et al. [5] transform the event logs into relationship matrices and find sudden drifts with change point detection algorithms, and Ostovar et al. [6] describes the sudden drift detection algorithm that relies on discovering a number of process trees from the event log and the calculation of the number of change operations to transform one tree into another. These papers focus on the identification of some specific drift types, limiting to sudden drifts and gradual drifts. These papers also do not provide an interpretable solution for visualizing the content of the drifts.

In this paper, we present a technique for process drift detection, called Visual Drift Detection (VDD). VDD extends existing techniques with the following features. First, our technique not only finds sudden drifts but also helps the user to interpret the four different types of drifts shown in Fig. 1. Second, it facilitates assessment of drifts through *visual interpretation* [8] by the help of an interactive visualization system. The Visual Drift Detection (VDD) system is built to explain input data on different levels of granularity and supports brushing and linking of the visualization views. Its back-end builds on the formal rigor of temporal logic of Declare constraints [9], [10] and time series analysis [11]. Key strengths of VDD are the clustering of declarative behavioral constraints that exhibit similar trends of change over time, the automatic detection of drift points, and the automated characterization of the drift types. We leverage this information about the trends in the data and represent the changes on the process behavior entailed by the drifts by means of enhanced Directly-Follows graphs [12], to provide further analysis features. These features allow us to detect and explain drifts that would otherwise go undetected by other techniques. We illustrate the usage of the VDD system on a real-world data set publicly available on the 4TU Data Centre.[1] The event log contains events from sepsis patients' pathways in the hospital [13]. We will henceforth refer to that data set as the Sepsis log.

This is a tool demonstration paper illustrating the new software implementation of the VDD system. The theoretical design and evaluations of the presented system have been partially described in [14], [15]. We remark that our earlier work did not include the advanced features we present here for drift type characterization and for the visualization of the entailed change on the process behavior.

II. The VDD Approach

Our technique takes an event log (henceforth, log for short) as an input and conducts a step-by-step visual analysis on process drifts. It consists of five steps, which we shall explain through the application of our tool on the case study of the Sepsis log. Figure 2 depicts the visualization system with connected views, showing the results of these steps.

1) Input and setting of parameters

In the first step the user provides an XES [16] and sets the parameters of the technique that will influence what can be observed. In particular, the *Win size* parameter determines the granularity of the drift analysis, and more specifically the number of traces that will be included in each time window. *Slide size* describes the number of traces that should be skipped to calculate the next window. The system offers hover-on explanations about each parameter. The in-depth analysis of the parameters is described in [14]. After that, the technique calculates the event log statistics and automatically proposes default parameters as shown in Fig. 2(h). Sepsis log has 1050 cases and 15 214 events with 16 event variants. We chose the *Win size* of 50, *Slide size* of 25, and *Cut threshold* of 420 for our analysis.

2) Window-based constraints mining and time series clustering

This is a preprocessing step for the visual analysis. We split the log into sub-logs. From each resulting part of the log, we measure the degree to which a set of behavioral relations in the form of declarative process constraints hold true in each window. In particular, we resort on the well-known declarative language DECLARE, whose full repertoire of constraints is described in [17]. The DECLARE constraints represent the behavior of a process by bind the occurrence of activities to the verification of certain conditions over other events in the trace. For example, PRECEDENCE(Release C, IV Liquid) states that IV Liquid can occur in the trace only if Release C occurred earlier. CHAINPRECEDENCE(Release C, Leucocytes) is a *chaining* constraint, which imposes that Leucocytes can occur only if Release C is the activity that occur immediately before it (i.e., no other activities can occur in between). NOTSUCCESSION(ER Registration, IV Liquid) is a *negative* constraint as it imposes that ER Registration *cannot* be followed by IV Liquid. For all constraints, we measure their support, confidence and interest factor. Based on established metrics of association rule mining [18], they indicate the extent to which the constraints are satisfied in the log traces. The detailed explanation of how those measures are computed is out of scope for this paper. For further information on that matter, the interested reader can refer to [10].

Specifically, the VDD system runs a background process to calculate the measures of DECLARE constraints and group the resulting time series into behavior clusters. First, traces in the log are sorted by the timestamp of their respective first events. Thereupon, we extract a sub-log of the given *Win size* from the first traces. We let the window slide over the log at the given *Slide size*. From each sub-log we mine the set of DECLARE constraints and compute their measures. In our case study, with the window size set to 50 and the sliding step to 25 we mine DECLARE constraints out of 41 sub-logs. For each sub-log, we compute the confidence of 3424 constraints. This step proceeds with the extraction of multi-variate time series that represent the trends of the constraints' confidence.

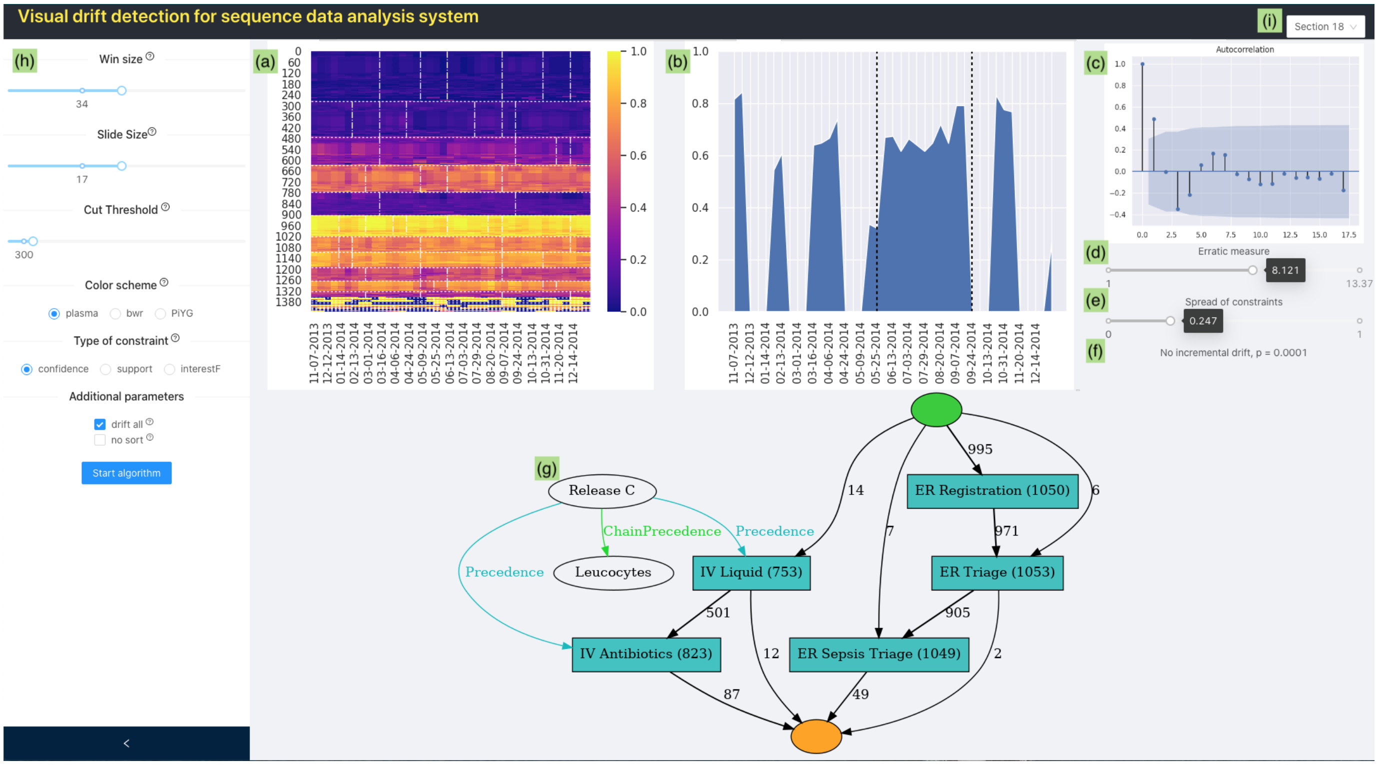
As a result of this step, we obtain numerous time series (one per constraint and measure) which we cluster into groups that exhibit similar confidence trends. Henceforth, we will refer to those groups as *behavior clusters*. In particular, we resort on hierarchical clustering [19] to find groups of constraints that exhibit similar confidence trends (henceforth, *behavior clusters*). Figure 2(a) shows the values of the time series (i.e., the confidence measures) through the plasma color-blind friendly color map [8], from blue (low peak) to yellow (high peak). The y-axis lists the constraints, the starting timestamp of the sub-logs lie on the x-axis. Constraints are sorted vertically by the similarity of their measures' trends. White dotted horizontal lines visually separate the behavior clusters. On the Sepsis data set, the Drift Map shows 18 behavior clusters.

3) Visualization of drifts

In this step, we detect change points in the set of time series, both for the whole log and each cluster separately. Those change points are what we identify as *drift points*. In the following, we will interchangeably name them as change or drift points depending on the context. We plot drift points in *Drift Maps* (Figure 2(a)) and *Drift Charts* (Figure 2(b)) to effectively communicate the drifts to the user.

The Drift Map shown in Fig. 2(a) illustrates the detected drift points over the time in the event log, which we shall collectively name as *drift situation*. We add vertical lines to mark such drift points. Drift Charts (e.g., those in Fig. 2(b)) have time on the x-axis and the average confidence of the constraints in a behavior cluster on the y-axis. We add vertical lines to denote drift points as in Drift Maps. In Fig. 2(b) we focus on behavior cluster 18 of the Sepsis log. We can observe

[1] https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460



Visual drift detection for sequence data analysis system

(h)

Win size: 34

Slide Size: 17

Cut Threshold: 300

Color scheme: ● plasma ○ bwr ○ PiYG

Type of constraint: ● confidence ○ support ○ interestF

Additional parameters

☑ drift all

□ no sort

Start algorithm

(i) Section 18

Erratic measure: 8.121

Spread of constraints: 0.247

No incremental drift, p = 0.0001

Figure 2: The user interface of the VDD system, running on the Sepsis event log [13]. (a) Drift Map. (b) Drift Chart. (c) Autocorrelation plot. (d) Erratic measure. (e) Spread of constraints view. (f) Incremental drifts test. (g) Extended Directly-Follows Graph. (i) Behavior cluster selection menu.

two drift points.

We also compute the values of measures called *spread of constraints* and *erratic measure* to quantify the extent of the drifting behavior [14]. The *spread of constraints* (shown in Fig. 2(e)) intuitively indicates how variable and subject to change the event log is. The measure ranges from 0 to 1: the more the behavior changes over time, the higher the value gets. In the Sepsis log, the measured spread of constraints is 0.247, which indicates a relatively small rate of change in the behavior. The *erratic measure* (shown in Fig. 2(d)) shows how a chosen cluster (Fig. 2(i)) compares to the cluster with the maximum degree of change in the same log.

4) Drift type detection

In this step, we use a range of methods to analyze drift types (as those shown in Fig. 1) and visualize them in the connected views. We use multi-variate time series change point detection algorithms to detect *sudden drifts*. In particular, we resort on the *Pruned Exact Linear Time (PELT)* algorithm [20] to detect change points in the whole multi-variate time series as well as within the behavior clusters. Thereupon, we make use of the stationarity analysis in ensemble with the visual inspection of Drift Charts to highlight *gradual* and *incremental drifts*. With the aid of autocorrelation plots, we seek for the behavior clusters exposing *reoccurring drifts*.

To show the results of this step, we resort on a mix of graphical and numerical representations: the aforementioned Drift Map together with Drift Charts, autocorrelation plots, and stationarity tests. In the chosen cluster 18, the system automatically identifies two *sudden drifts* as shown in the Drift Chart (Fig. 2(b)). To check for *incremental drifts*, we inspect the results of the stationarity test (shown in Fig. 2(f)). For the chosen behavior cluster, the VDD system reports no incremental drift. Figure 2(c) depicts an autocorrelation plot that shows how the time series correlates with itself with a step defined in the y-axis. The blue area on this plot shows the significant region of the analysis. Cluster 18 reveals an autocorrelation on step 2, meaning that the drift shows signs of seasonality – thus being classifiable as a *reoccurring drift*.

5) Understanding the drift behavior

To get an understanding of the effect of drifts on the process behavior, we visually represent the general behavior found in the log extended with specific behavior shown in a chosen behavior cluster. In particular, we use the gathered information on the measured DECLARE constraints in a behavior cluster and draw it on top of Directly-Follows graphs [12] such as the one in Fig. 2(g). A Directly-Follows graph connects via arcs the activities (nodes) with those other activities that followed at least once in a trace. Arcs are weighted by the number of such sequences. Nodes are weighted by the frequency with which the related activities occur in the log. The Directly-Follows graph depicts the behavior that is common to the entire event log. We add arcs highlighted with different colors that represent additional Declare, cluster-specific constraints. Negative Declare constraints are colored in red. Chaining constraints are in green. All other relationships are in blue. For cluster 18 we see from Fig. 2(g) that activities Release C and Leucocytes occur in sequence, bound by the ChainPrecedence(Release C, Leucocytes) constraint. Furthermore, Precedence(Release C, IV Liquid) and Precedence(Release C, IV Antibiotics) suggest that IV Liquid and IV Antibiotics require Release C to occur before, unlike in the general behavior.

III. Maturity, Documentation and Screencast

We implemented the VDD system as a Python-based standalone program for command line execution, and as a web application with back-end and front-end parts. The algorithms are implemented using Python 3, resorting on the *scipy* library for time-series clustering and on the *ruptures* library for change point identification. We use PM4Py[2] [21] for the Directly-Follows Graph visualization. We use the MINERful[3] Java package for the discovery and measuring of Declare constraints [10]. The front-end of the tool is implemented with the *React JavaScript* library. The back-end is implemented with *flask* python library. We run our experiments using a laptop equipped with an Intel Core i5 at 2.40GHz × 2 with 8GB of RAM. With this modest hardware, the tool was able to process data and produce the analysis outcome in about 17 seconds using a real-size event log with 15 214 events from 16 activities over 1050 traces. This indicates that the VDD system has reached a fairly large degree of maturity as it performs well in terms of scalability.

We have created a project website for the VDD system, from which it can be downloaded together with its sources at https://github.com/yesanton/Process-Drift-Visualization-With-Declare. 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. The web tool with a graphical interface is also available at https://yesanton.github.io/driftvis, to be used for testing without the need to install the software on a local machine. A screencast documenting its usage is available at https://youtu.be/mHOgVBZ4Imc. The GitHub project page contains the step by step tutorial of how to use the web-based tool. It is available at https://github.com/yesanton/Process-Drift-Visualization-With-Declare/blob/master/publications/icpm-2020-demo-tutorial.pdf

In future work, we will focus on the prediction of drifts in running processes and the improvements of the interactivity of the visualization system. Furthermore, we will conduct user studies to assess the perceived quality of the tool.

Acknowledgements.

This work is partially funded by the EU H2020 program under MSCA-RISE agreement 645751 (RISE_BPM). Artem Polyvyanyy is partly supported by the Australian Research Council Discovery Project DP180102839. Claudio Di Ciccio is partly supported by the MIUR under grant "Dipartimenti di eccellenza 2018-2022" of the Department of Computer Science of Sapienza University of Rome. Anton Yeshchenko thanks Maryna Zadoianchuk and Oleksii Tkachenko for their assistance during the development of the web application.

References

[1] W. M. P. van der Aalst, *Process Mining - Data Science in Action*. Springer, 2016.

[2] M. L. van Eck, X. Lu, S. J. J. Leemans, and W. M. P. van der Aalst, "PM ˆ2 : A process mining project methodology," in *CAiSE*. Springer, 2015, pp. 297–313.

[3] R. Moreno and R. E. Mayer, "Visual presentations in multimedia learning: Conditions that overload visual working memory," in *VISUAL*, D. P. Huijsmans and A. W. M. Smeulders, Eds. Springer, 1999, pp. 793–800.

[4] A. Maaradji, M. Dumas, M. La Rosa, and A. Ostovar, "Detecting sudden and gradual drifts in business processes from execution traces," *IEEE TKDE*, vol. 29, no. 10, pp. 2140–2154, 2017.

[5] C. Zheng, L. Wen, and J. Wang, "Detecting process concept drifts from event logs," in *OTM*. Springer, 2017, pp. 524–542.

[6] A. Ostovar, S. J. J. Leemans, and M. L. Rosa, "Robust drift characterization from event streams of business processes," *ACM Trans. Knowl. Discov. Data*, vol. 14, no. 3, pp. 30:1–30:57, 2020. [Online]. Available: https://doi.org/10.1145/3375398

[7] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, 2014.

[8] C. Ware, *Information visualization: perception for design*. Elsevier, 2012.

[9] W. M. P. van der Aalst and M. Pesic, "DecSerFlow: Towards a truly declarative service flow language," in *WS-FM*, ser. Lecture Notes in Computer Science, vol. 4184. Springer, 2006, pp. 1–23.

[10] C. Di Ciccio and M. Mecella, "On the discovery of declarative control flows for artful processes," *ACM TMIS*, vol. 5, no. 4, pp. 24:1–24:37, 2015.

[11] G. C. Reinsel, *Elements of multivariate time series analysis*. Springer, 1993.

[12] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Discovering block-structured process models from event logs - A constructive approach," in *PETRI NETS*. Springer, 2013, pp. 311–329.

[13] F. Mannhardt and D. Blinde, "Analyzing the trajectories of patients with sepsis using process mining," in *BPMDS/EMMSAD*. CEUR-WS.org, 2017, pp. 72–80.

[14] A. Yeshchenko, C. Di Ciccio, J. Mendling, and A. Polyvyanyy, "Comprehensive process drift detection with visual analytics," in *ER*. Springer, 2019, in print.

[15] A. Yeshchenko, C. D. Ciccio, J. Mendling, and A. Polyvyanyy, "Comprehensive process drift analysis with the visual drift detection tool," in *ER Demos*. CEUR-WS.org, 2019, pp. 108–112.

[16] "IEEE standard for extensible event stream (xes) for achieving interoperability in event logs and event streams," pp. 1–50, Nov 2016. [Online]. Available: http://dx.doi.org/10.1109/IEEESTD.2016.7740858

[17] W. M. P. van der Aalst and M. Pesic, "DecSerFlow: Towards a truly declarative service flow language," in *WS-FM*. Springer, 2006, pp. 1–23.

[18] J. Adamo, *Data mining for association rules and sequential patterns - sequential and parallel algorithms*, J. Adamo, Ed. Springer New York, 2001.

[19] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, "Time-series clustering - a decade review," *IS*, vol. 53, no. C, pp. 16–38, Oct. 2015.

[20] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012.

[21] A. Berti, S. J. van Zelst, and W. M. P. van der Aalst, "Process mining for python (pm4py): Bridging the gap between process- and data science," *CoRR*, vol. abs/1905.06169, 2019.

[2] http://pm4py.org, https://github.com/pm4py

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