Detecting Change in Processes Using Comparative Trace Clustering

B.F.A. Hompes[1,2], J.C.A.M. Buijs[1], W.M.P. van der Aalst[1], P.M. Dixit[1,2], and J. Buurman[2]

[1] Department of Mathematics and Computer Science
Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Philips Research, Eindhoven, The Netherlands
{b.f.a.hompes,h.m.w.verbeek,w.m.p.v.d.aalst}@tue.nl
{prabhakar.dixit,hans.buurman}@philips.com

Abstract. Real-life *business processes* are complex and show a high degree of variability. Additionally, due to changing conditions and circumstances, these processes continuously evolve over time. For example, in the healthcare domain, advances in medicine trigger changes in diagnoses and treatment processes. Besides changes over time, case data (e.g. treating physician, patient age) also influence how processes are executed. Existing *process mining* techniques assume processes to be static and therefore are less suited for the analysis of contemporary flexible business processes. This paper presents a novel *comparative trace clustering* approach that is able to expose changes in behavior. Valuable insights can be gained and process improvements can be made by finding those points in time where behavior changed and the reasons why. Evaluation on real-life event data shows our technique can provide these insights.

Keywords: process mining, trace clustering, concept drift, process comparison

1 Introduction

Business processes in flexible environments typically depend on many factors. Due to changing conditions and circumstances, these processes continuously evolve over time. For example, advances in medicine can change how patients are treated or how diagnoses are made in hospitals. Other changing circumstances could be new legislation, seasonal effects or even preferences of involved resources. As a result, in these flexible types of processes many cases follow a unique path through the process. This variability causes problems for existing process mining techniques since processes are assumed to be structured and in a steady state. Contemporary process mining techniques return spaghettilike processes and potentially misleading results when this is not the case [3, 7, 16]. The discovered process models capture behavior possible at any given point in time. Often, however, the process context changes and what was possible before is no longer possible or vice-versa. Figure 1 shows the relevance of the problem with an example process that has changed over time.

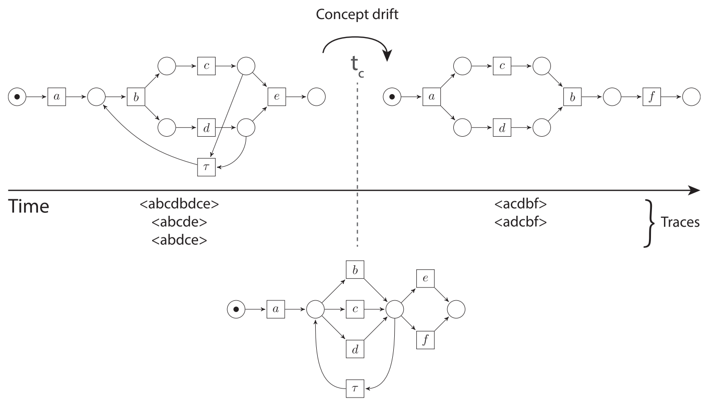

Fig. 1: Demonstration of concept drift in a simple process. Techniques that consider the entire event log can return misleading results.

This so-called *concept drift* is one of the key challenges in process mining, as discussed in the Process Mining Manifesto [2]. Few techniques have been proposed to deal with concept drift in a business process setting [5, 6, 9, 11, 17]. Where existing techniques focus mainly on change in control-flow, our focus is on detecting changing behavior including data aspects as well.

Finding changes in behavior can be of great value to process owners. Detecting unwanted changes, for example, can identify potential risks while positive change can lead to perdurable process improvement. Techniques that consider the entire event log at once cannot show the individual changes that occur throughout the lifetime of a process [5]. For example, specific types of behavior might occur for a limited time only, or can have a seasonal nature. Behavior can also merge with, or emerge from other behavior.

Analysis of differences in behavior is supported by comparing clusterings created for two selected partitions of the event log. This way, we can not only compare differences in behavior over time, but behavior for different case groups (e.g. different age groups, customer types, cases handled by different resources) can be compared as well. As such, the analysis of differences in process behavior is generalized.

Recently we proposed a novel technique for the detection of common and deviating behavior using *trace clustering* [8]. The process context is considered by taking both control-flow as well as case and event data into account. Hence, clustering of cases is not limited to finding similar execution paths. The trace clustering technique is based on the Markov cluster (MCL) algorithm, which is a fast and scalable cluster algorithm for graphs [15]. It is able to autonomously discover a (non-specified) number of clusters of different sizes and densities. Hence, the resulting clustering shows both mainstream and deviating behavior. The technique can be used to exploratively analyse the event data at hand in order to gain novel insights into the underlying process(es).

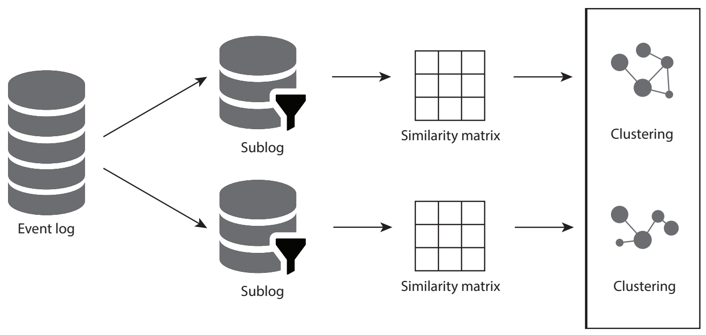

Fig. 2: A graphical overview of the approach. Differences in behavior can be compared using comparative trace clustering.

In this paper, we extend on the work in [8] by providing a new technique for *change point detection* and a means to compare clusterings. Figure 2 shows a high-level overview of our approach. By incorporating the time dimension we can discover temporal evolutions in process behavior. Changes in behavior causing or caused by differences in case data can therefor be detected as well. In order to detect change points, we look at similarities between cases. We consider the effect of new events on a clustering of active cases.

The remainder of this paper is organized as follows. Section 2 briefly introduces necessary preliminary definitions. The detection of change points is described in Section 3. Section 4 describes the comparative trace clustering approach. An experimental evaluation on two real-life event logs is performed in Section 5. Section 6 discusses related work. The paper concludes with a summary and planned future work in Section 7.

2 Preliminaries

Typically, the executed *events* of multiple *cases* of a *process* are recorded in an *event log*. An event is an execution of an activity potentially having additional data attributes such as a timestamp or the responsible resource. A trace is a finite sequence of events, and describes one specific instance (i.e. a case) of the process at hand in terms of the executed events. A case can also have additional (case-wide) attributes such as a patient birthdate or diagnosis result. Definitions in this section are based on those as defined in [1].

Definition 1 (Event, attribute). *Let $\mathcal{E}$ be the event universe, i.e. the set of all possible event identifiers. Events may be characterized by various attributes, e.g. an event may have a timestamp, correspond to an activity, be executued by a particular person, etc. Let N be a set of attribute names. For any event $e \in \mathcal{E}$ and attribute name $n \in N$: $n(e)$ is the value of attribute n for event e. If event e does not have an attribute named n, then $n(e) = \perp$ (null value).*

Typically, the following attributes are present in all events: *activity*(e) is the *activity* associated to event e, *time*(e) is the *timestamp* of e, and *resource*(e) is the *resource* associated to e. Additional event attributes can be the cost associated with the event, the outcome of an activity (e.g. surgery result), etc.

Definition 2 (Case, trace, event log). *Let $\mathcal{C}$ be the case universe, i.e. the set of all possible case identifiers. Cases, like events, have attributes. For any case $c \in \mathcal{C}$ and attribute name $n \in N$: $n(c)$ is the value of attribute n for case c ($n(c) = \perp$ if c has no attribute named n). Each case has a mandatory attribute 'trace': $trace(c) \in \mathcal{E}^*$. $\hat{c} = trace(c)$ is a shorthand notation for referring to the trace of a case. A trace is a finite sequence of events $\sigma \in \mathcal{E}^*$, such that each event appears only once, i.e. for $1 \leq i \leq \ j \leq |\sigma|$: $\sigma(i) \neq \sigma(j)$. For any sequence $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_n \rangle$, $set(\sigma) = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ converts a sequence into a set, e.g. $set(\langle a, b, c, b, c, d \rangle) = \{a, b, c, d\}$. An* event log *is a set of cases $L \subseteq \mathcal{C}$ such that each event appears at most once in the entire log, i.e. for any $c, c' \in L$ such that $c \neq c'$: $set(\hat{c}) \cap set(\hat{c}') = \emptyset$.*

In the example traces in Figures 1 and 3 a simplified form is used where events are represented solely by the activities they execute. In this form, a trace is a sequence of activities and an event log a multiset of traces (since in this simplified form cases can share their trace).

Definition 3 (Trace Similarity Matrix). *Let $L \subseteq \mathcal{C}$ be an event log. $\mathcal{M}(L) = (L \times L) \rightarrow [0.0, 1.0]$ denotes the set of all possible trace similarity matrices over L. For cases $c, c' \in L$ and a trace similarity matrix $M \in \mathcal{M}(L)$, $M(c, c')$ denotes the similarity between c and c'. $\sum M = \sum_{c,c'} M(c, c')$ denotes the sum of all elements in M. Furthermore, standard matrix operations such as addition, subtraction, multiplication, etc. are supported as well.*

Definition 4 (Trace Clustering). *Let $L \subseteq C$ be an event log. A trace cluster over L is a subset of L. A trace clustering $TC \subseteq \mathcal{P}(L)$*[1] *is a set of trace clusters over L. We assume every case to be part of at least one trace cluster, i.e. $\bigcup \ TC \ = \ L$. Cases can be in multiple clusters, i.e. cluster overlap is allowed.*

3 Detecting Change in Behavior

Discovering a process model on an entire real-life event log will often lead to a spaghetti model since it has to represent all past traces [3, 7, 16]. Similarly, clustering the entire event log will show groups of behavior that were possible at any given point in time. The evolution of this behavior is not shown.

Our technique is based on the technique proposed in [8] where trace clustering and outlier detection are combined in order to find mainstream and deviating behavior. It relies on the Markov cluster (MCL) algorithm [15] to find groups of cases (trace clusters) that share behavior on a set of selected perspectives. By incorporating both control-flow and case data, the process context is taken

[1] $\mathcal{P}(L)$ denotes the powerset over event log L, i.e. all possible sublogs of L.

into account. Because MCL is neither biased towards globular or local clusters and is able to find clusters of different density, we can distinguish common and exceptional behavior based on cluster sizes. As the number of clusters is discovered rather than set beforehand, changes in behavior will be reflected in change in the clustering.

MCL uses a stochastic similarity matrix between cases as its input. It alternates an expansion step that raises the matrix to a given power with the inflation step. Inflation raises each element to a given power and normalizes the matrix such that it is stochastic again. This alternation eventually results in the separation of the matrix into different components which is interpreted as a clustering. Similarities between cases are calculated by mapping cases to profile vectors using a set of selected perspectives and computing pair-wise vector similarity values. Possible perspectives are the occurrence or frequency of certain activities, the values for case or event data, etc. For more information on the calculation of this similarity matrix and the use of the MCL algorithm in trace clustering we refer the reader to [8].

We propose to utilize the similarity matrix between cases to detect changes in behavior. An overview of this approach can be seen in Figure 3, where five example (simple) traces are drawn over time. Over time, the occurrence of new events in a certain trace will change the similarity of that trace to other traces, which is reflected in the similarity matrix. As this matrix is the input for the clustering algorithm, the impact on the similarity matrix is a good indicator for how much the clustering will change. Thus, in order to detect potentially interesting change points, we compute the change in the values of the similarity matrix over time. Similar to the approaches that use statistical tests [5, 10, 11], a large difference indicates a significant change in behavior. For each event window, cases that have events in or before that window are considered in the calculation of the next similarity matrix. Events occurring after the current event window are not considered. As such, for each case, the prefix of its trace is taken. Different window sizes can be considered. Case attributes (e.g. patient age or gender) are considered to be known from the very first event in the trace of a case. As a result, when looking at case-wide data, change points will reflect on the first event of a deviating case. This projection of cases leads to a sublog containing all events up to a certain time. A similarity matrix is calculated for that sublog and compared with the previous similarity matrix. If change points have been identified, clusterings of cases occurring before and after these points can be created to compare behavior before and after the change point.

For example, take the traces depicted in Figure 3, and consider cases to be similar when they share activities. At first, up to event window W_3, trace t_4 seems to be quite similar to the other three traces seen so far. However, in window W_4 this changes due to activity f which has replaced activity e. This change is reflected in the similarity matrices M_1 to M_5. Consequently the value for $\varDelta_3 = |\sum(M_4 - M_3)|$ will be significant and indicates a possible change point in W_4. Since the range for the values for $|\sum(M_x - M_{x-1})|$ for future windows is unknown, we normalize all delta values after each measurement.

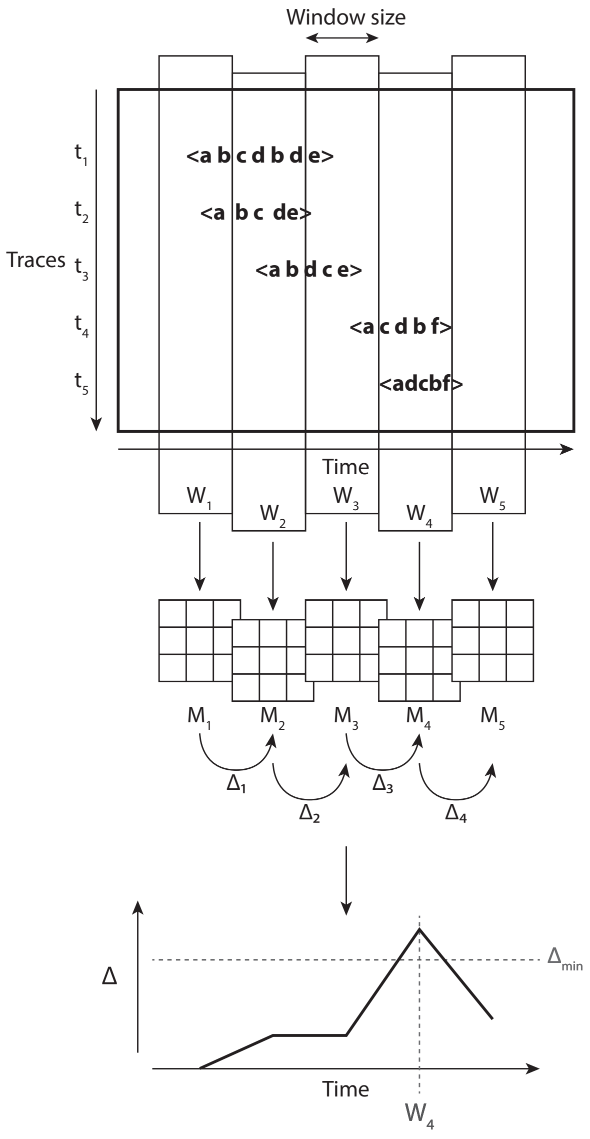

Fig. 3: A graphical overview of the change detection approach. In order to detect changes in behavior, the difference in similarity matrices over time is calculated. In this example, significant change occurred in window W_4.

4 Comparative Clustering

In order to analyse the effect of changing behavior in a process, different clusterings can be created and compared both programmatically and visually. By comparing clusterings created for different sublogs (e.g. before and after an identified change point), we can see where behavior changed and why. As discussed, it is also possible to manually create selections of cases based on time or data attributes in order to compare behavior. This can be used to compare behavior for different age groups, for patients from different geographical locations, etc.

Because of the properties of the MCL algorithm discussed in Section 3, change is reflected in the cluster structure. For example, cluster sizes indicate the frequency of the captured behavior. Behavior that used to be common but is becoming less and less frequent will still be clustered separately rather than being merged with (different) common behavior, as long as the behavior remains dissimilar enough.

Clusters of cases can be annotated with descriptions about the cases that are present. Shared activities between traces and similar data values are a few of the possibilities. For example, a cluster might group cases of patients that all share a certain diagnosis. This diagnosis can be used to describe the cluster.

Two example clusterings are shown in Figure 4. In the left clustering, one cluster is selected. By highlighting those clusters in the right clustering that share behavior with the selected cluster(s) on the left we can show how behavior has changed. It is possible to see how cases are clustered or what annotations are shared. For example, we might compare two years of patient data, clustered on diagnosis. Patients that are present in both years are highlighted as shared cases (in dark gray), while clusters that share some or all diagnoses (shared annotations) are highlighted (in light gray). Within a trace clustering, clusters that share behavior are connected. As such, when comparing two clusterings, clusters that are split into or have emerged from multiple clusters can be found. This can indicate that behavior has become more specific or more general.

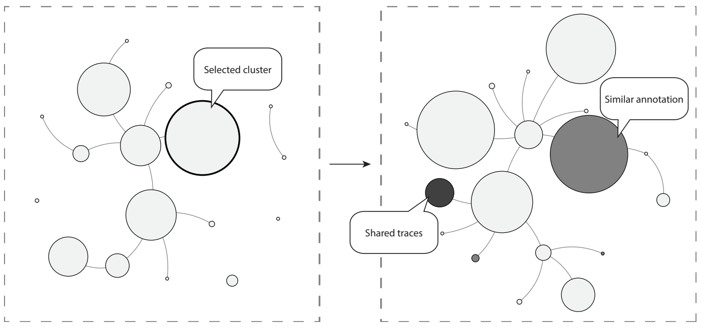

Fig. 4: Two clusterings compared. Highlighted clusters on the right show how behavior on the left has changed.

5 Evaluation

In order to evaluate the approach we use two real-life event logs. The first log comes from a Dutch academic hospital that contains cases pertaining to cancer treatment procedures. It was originally used in the first Business Process Intelligence Contest (BPIC 2011) [13]. The second event log contains cases of building permit applications provided by a Dutch municipality. This log is part of the 2015 edition of the BPI Challenge (BPIC 2015) [14]. These event logs were used so that the results can be reproduced. Our technique has been implemented in the process mining tool ProM[1], and is publicly available in the *TraceClustering* package. In the results shown here, we used parameters expansion=2, inflation=15 for the MCL algorithm.

5.1 Hospital event log

The first event log contains cases of different stages of malignancy and of different parts of the body. Also, information is present about the diagnosis, treatment, specialism required, patient age, organisational group (hospital department), etc. This log contains 1,143 cases, 150,291 events and 624 distinct activities. There are 981 different executions paths (activity sequences). There are many attributes present on both the event and case level. All of these attributes can obtain several different values, leading to a large heterogeneity in the log. As cases are recorded between January 2005 and March 2008, the event log is likely to exhibit drifts.

For each case, there are 16 attributes for 'diagnosis code', referring to the diagnoses the patient received for different parts of their body. By clustering on these attributes and comparing the years 2005 and 2006 we can see trends in common and exceptional diagnoses. Figure 5 shows how this diagnosis has evolved. The clustering on the left represents behavior in 2005 whereas the clustering on the right represents 2006. Patients that were in the selected cluster and have had activities in both years are highlighted in dark gray. Groups of patients that have had (partially) shared diagnoses are marked light gray. For example, out of the 539 cases in 2015, 78 (14.5%) were diagnosed with codes M13, 822, and 106, on different parts of the body. Out of the 765 cases in 2016, only 84 (11.0%) had a diagnosis using the same codes. What's more, is that these diagnoses are present for more body parts as well. This could indicate a trend in diseases or be due to an improvement in diagnosis detail. As there are many smaller clusters in 2006 that have additional diagnoses (light gray), we can deduce that for the selected diagnosis, the related diagnoses have become more specific and diagnoses are also made on other parts of the body. In Figure 5 process maps and differences in activities are shown for two highlighted clusters.

Besides diagnosis codes, every case has 16 possible attributes for 'treatment code', referring to the treatments the patient received on different parts of their body. This leads to many possible treatment combinations for different diagnoses. We compare the clustering created for the combined attributes diagnosis code

[1] See `http://promtools.org`

and treatment code for the years 2005 and 2006. Using our technique, different clusters are discovered, each cluster contains specific combinations of diagnosis and treatment. By comparing the results for the two years, changes in treatments for specific diagnoses become visible. As we can see from Figure 6, treatment for some diagnoses have changed. Again cases in 2005 are shown on the left and cases in 2006 are shown on the right. We can see that there are two clusters that contain patients that were active in 2005, one of which is much smaller than the other. For the larger cluster, additional diagnoses were made and additional treatments were performed. As a result, more cases share this behavior. The call-outs in Figure 6 again show differences in activities between the two years.

These differences indicate that, over time, treatments for certain diagnoses have changed. Considering the type of process, this could be due to specific patient needs, changes in protocols or advances in medicine. A likely reason is that diagnoses were made (or recorded) with greater detail in 2006 versus in 2005. Insights such as these can be gained easily and can be used to verify or specify protocols, check whether certain behavior is changing or for auditing purposes.

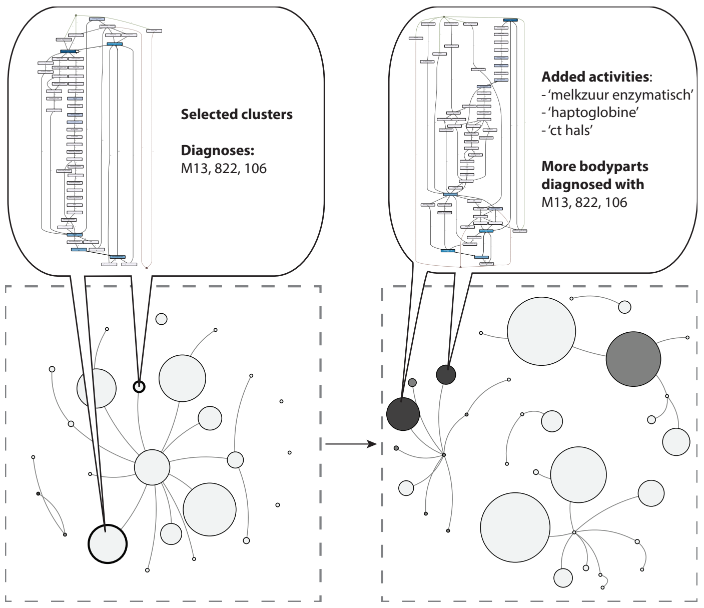

Fig. 5: Hospital log clustered on diagnosis code for cases active in 2005 (left) and 2006 (right). Changes in diagnoses are discovered. More bodyparts are diagnosed with codes M13, 822 and 106.

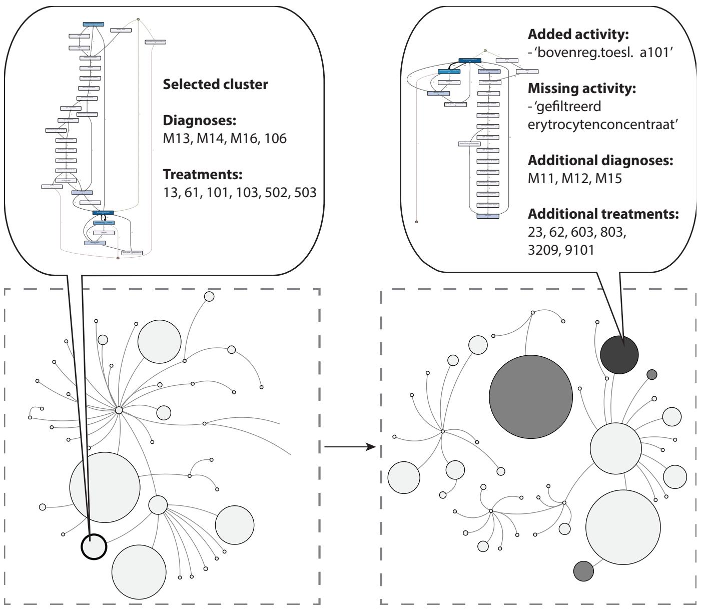

Fig. 6: Hospital log clustered on diagnosis code and treatment code in 2005 (left) and 2006 (right). Treatments for specific diagnoses are changing over time. Additional diagnoses and treatments are found.

5.2 Municipality event log

The second event log contains cases of building permit applications in a Dutch municipality. Information is present about the type of permit, the costs associated with the permit, the involved resources, etc. Again, each attribute can have several different values. This log contains 1,199 cases recorded between late 2010 and early 2015 with in total 52,217 events and 398 distinct activities. As there are 1170 different execution paths, almost all cases are unique from the control-flow perspective.

Each case has an attribute 'parts' that refers to the different permit types that are involved in the case it describes. As a pre-processing step, values containing multiple types were split up over different case attributes, one for each type. Besides permit types, each case is also labeled with the attribute 'term name', describing which status has been assigned to the permit application. Possible values are 'permit granted', 'additional information required', 'term objection and appeal', etc.

We cluster the cases in the log on both permit type and term name and compare cases in 2011 with cases in 2012. The results are shown in Figure 7. As we can see, few clusters are discovered, indicating only slight differences in behavior on these perspectives. A group of cases pertaining to mainly construction and environmental permits that are in the 'objection and appeal' term is selected in the 2011 clustering. Over a one-year period, most of this behavior has merged

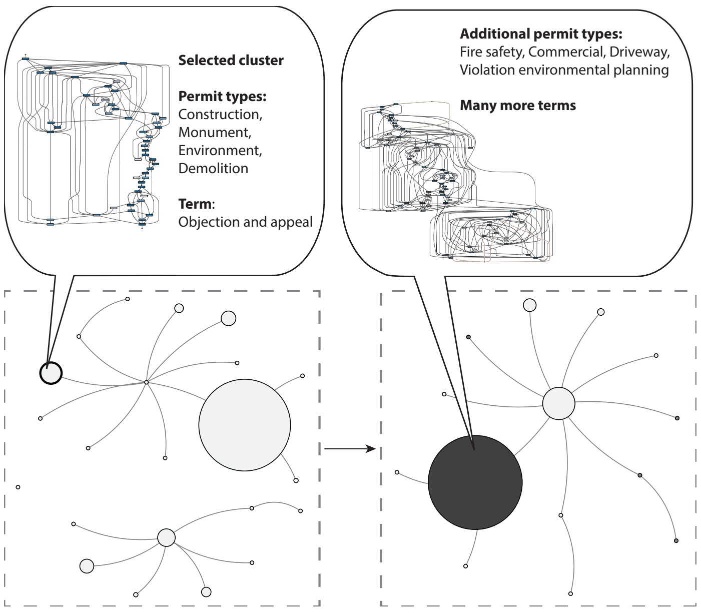

Fig. 7: Municipality log clustered on permit type and term description. In this case, the discovered change in behavior leads to limited insights.

with the biggest group of cases, which now represents almost all behavior in the log in 2012.

Though there clearly are differences in behavior between the two years, it is difficult to interpret these findings directly. More analysis is necessary in order to discover results that are meaningful in a business process management context. For example, the activities that are executed could be considered when clustering the cases in the log. As such, clusters would show differences and similarities in control-flow as well. This indicates that the choice for the clustering perspectives is essential in finding valuable insights about differences in behavior.

6 Related Work

Although concept drift is a well-studied topic in the data mining and machine learning communities, little work has been done on detecting concept drift in business processes. Bose et al. were the first to consider concept drift and change detection in a process mining setting [5]. In their work, a classification of possible changes in business processes is given, and statistical hypothesis tests are used to detect regions of change. Even tough the authors consider the possibility of change in data attributes, the scope of their work is limited to the detection of control-flow changes in a process manifested as sudden drifts over a period of time. More recently, Martjushev et al. built on this work by looking at gradual and multi-order dynamics to detect concept drift in control-flow [11]. They extend the work in [5] by providing solutions to detect gradual change as well. By considering multi-order dynamics through the use of an adaptive window technique, process change occurring at multiple levels of mixed time granularity can be detected. Maaradji et al. employ statistical tests over the distributions of runs observed in two consecutive time windows in order to detect concept drift [10]. As noted by the authors, in order to find differences in process behavior a notion of equivalence is necessary. In their paper, a notion of run-equivalence is used. It is shown that drift can be identified fast and accurately by using an adaptive sliding window technique. As a result, it can be used in an online (streaming) setting as an oracle as to when a discovered model should be updated.

Weber et al. employ probabilistic deterministic finite automata (PDFA) to represent the probability distributions generated by process models [17]. Similar to [5] and [10], statistical hypothesis tests are used to detect whether or not a distribution has changed significantly from a ground truth. The aim of their technique is to identify process change as soon as possible, but with confidence that change is significant, in order to discover a model representing reality as good as possible. As such, only drift in control-flow is considered. In [6] a different technique is proposed to automatically detect and manage concept drift in an online setting. Here, concept drift is detected real-time using an estimation technique based on abstract interpretation of the process and sequential sampling of the log. The fitness of prefixes of new samples taken from the log is checked against that of prefixes of initial samples. A change point is identified when there is a significant difference between these two points. In the above-mentioned techniques however, data attributes are not considered. As such, only changes in control-flow behavior can be discovered.

Trace clustering techniques are often used to find different process variants. Several trace clustering techniques have been proposed in the field of process mining, and an extensive comparative analysis of trace clustering techniques has recently been performed in [12]. Often, however, the temporal dimension is not considered. In [9], the starting time of each process instance is used as an additional feature in trace clustering. By combining control-flow and time features, the clusters formed share both a structural similarity and a temporal proximity. The technique is based on the technique proposed in [4] and considers different types of changes, including sudden, recurring, gradual, and incremental changes. In more complex evolving business processes however, including the temporal proximity of cases might lead to misleading results. For example when seasonal drifts are intertwined with gradual changes in the process.

The technique proposed in this paper uses similar ideas and concepts as used in the papers mentioned above. However, trace clustering is used to find common and deviating process behavior. By looking at both control-flow as well as data attributes, the technique is made context-aware. We extend the technique in [8] by including change detection in behavioral similarities between cases. In this way, we can identify changes in common and deviating behavior, on both the control-flow and data perspectives.

7 Conclusions and Future Work

Real-life *business processes* are often complex while exhibiting a high degree of variability. Due to changing conditions and circumstances, these processes continuously evolve over time. Existing *process mining* techniques assume the process to be static and therefore are less suited for the analysis of contemporary business processes. In this paper we presented a novel *comparative trace clustering* approach that is able to expose differences in behavior in a process. By using both control-flow and case data we take the process context into account. Insights can be gained into how and why behavior has changed by comparing changes in clusterings over different partitions of the log. This information can then be used for further analysis, e.g. to design protocols, for early detection of unwanted behavior or for auditing purposes. Besides the time dimension, different data and control-flow attributes can be utilized in order to distinguish groups of behavior.

While our initial results show that indeed promising insights can be achieved, there is still quite some manual work involved. More work is needed to further automate the analysis process. In the future we would also like to look into how changes in process behavior can be analysed in an online setting. Different ways to visualize change in behavior should be explored as well.

Bibliography

[1] W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, Berlin, 2011.

[2] W.M.P. van der Aalst, A. Adriansyah, A.K.A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, R.P.J.C. Bose, P. van den Brand, R. Brandtjen, J.C.A.M. Buijs, et al. Process Mining Manifesto. In *Business Process Management Workshops*, pages 169–194. Springer, 2012.

[3] R.P.J.C. Bose and W.M.P. van der Aalst. Context Aware Trace Clustering: Towards Improving Process Mining Results. In *Proceedings of the SIAM International Conference on Data Mining*, pages 401–412. Society for Industrial and Applied Mathematics, 2009.

[4] R.P.J.C. Bose and W.M.P. van der Aalst. Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models. In *Business Process Management Workshops*, pages 170–181. Springer, 2010.

[5] R.P.J.C. Bose, W.M.P. van der Aalst, I. Žliobaitė, and M. Pechenizkiy. Handling Concept Drift in Process Mining. In *Advanced Information Systems Engineering*, pages 391–405. Springer, 2011.

[6] J. Carmona and R. Gavalda. Online Techniques for Dealing with Concept Drift in Process Mining. In *Advances in Intelligent Data Analysis XI*, pages 90–102. Springer, 2012.

[7] S. Goedertier, J. De Weerdt, D. Martens, J. Vanthienen, and B. Baesens. Process Discovery in Event Logs: An Application in the Telecom Industry. *Applied Soft Computing*, 11(2):1697–1710, 2011.

[8] B.F.A. Hompes, J.C.A.M. Buijs, W.M.P. van der Aalst, P.M. Dixit, and J. Buurman. Discovering Deviating Cases and Process Variants Using Trace Clustering. In *Proceedings of the 27th Benelux Conference on Artificial Intelligence (BNAIC), November 5-6, Hasselt, Belgium*, 11 2015.

[9] D. Luengo and M. Sepúlveda. Applying Clustering in Process Mining to Find Different Versions of a Business Process that Changes over Time. In *Business Process Management Workshops*, pages 153–158. Springer, 2012.

[10] A. Maaradji, M. Dumas, M. La Rosa, and A. Ostovar. Fast and Accurate Business Process Drift Detection. In *Business Process Management*, pages 406–422. Springer, 2015.

[11] J. Martjushev, R.P.J.C. Bose, and W.M.P. van der Aalst. Change Point Detection and Dealing with Gradual and Multi-order Dynamics in Process Mining. In *Perspectives in Business Informatics Research*, pages 161–178. Springer, 2015.

[12] T. Thaler, S.F. Ternis, P. Fettke, and P. Loos. A Comparative Analysis of Process Instance Cluster Techniques. In *Proceedings of the 12th International Conference on Wirtschaftsinformatik. Internationale Tagung Wirtschaftsinformatik (WI-15), March 3-5, Osnabrck, Germany.* Universitt Osnabrck, 3 2015.

[13] B.F. van Dongen. Real-life Event Logs - Hospital Log, 2011. URL: `http://dx.doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54`, `doi:10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54`.

[14] B.F. van Dongen. BPI Challenge 2015, 2015. URL: `http://dx.doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1`, `doi:10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1`.

[15] S. Van Dongen. A Cluster Algorithm for Graphs. Technical report, National Research Institute for Mathematics and Computer Science in the Netherlands, 2000.

[16] G.M. Veiga and D.R. Ferreira. Understanding Spaghetti Models with Sequence Clustering for ProM. In *Business Process Management Workshops*, pages 92–103. Springer, 2010.

[17] P. Weber, B. Bordbar, and P. Tino. Real-Time Detection of Process Change using Process Mining. In *Imperial College Computing Student Workshop*, pages 108–114, 2011.