Event-Log Abstraction using Batch Session Identification and Clustering

Massimiliano de Leoni
Department of Mathematics - University of Padua
Padua, Italy
deleoni@math.unipd.it

Safa Dündar
Micro Focus
Utrecht, The Netherlands
safa.dundar@microfocus.com

ABSTRACT

Process-Mining techniques aim to use event data about past executions to gain insight into how processes are executed. While these techniques are proven to be very valuable, they are less successful to reach their goal if the process is flexible and, hence, it exhibits an extremely large number of variants. Furthermore, information systems can record events at very low level, which do not match the high-level concepts known at business level. Without abstracting sequences of events to high-level concepts, the results of applying process mining (to, e.g., discover a model) easily become very complex and difficult to interpret, which ultimately means that they are of little use. A large body of research exists on event abstraction but typically a large amount of domain knowledge is required, which is often not readily available. Other abstraction techniques are unsupervised, which ultimately return less accurate results and/or rely on stronger assumptions. This paper puts forward a technique that requires limited domain knowledge that can be easily provided. Traces are divided in batch sessions, and each session is abstracted as one single high-level activity execution. The abstraction is based on a combination of automatic clustering and visualization methods. The technique was assessed on two case studies about processes characterized by high variability. The results clearly illustrate the benefits of the abstraction to convey accurate knowledge to stakeholders.

KEYWORDS

Process Discovery, Event Log Abstraction, Clustering, Flexible Processes, Visual Analytics

ACM Reference Format:
Massimiliano de Leoni and Safa Dündar. 2020. Event-Log Abstraction using Batch Session Identification and Clustering. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3341105.3373861

1 INTRODUCTION

Nowadays, large, complex organizations leverage on well-defined processes to try to carry on their business more effectively and efficiently than their competitors. In a highly competitive world, organizations aim to continuously improve their business performance, which ultimately boils down to improving their process.

The first step towards improvement is to understand how processes are actually being executed. The understanding of the actual process enactment is the goal of *Process Mining* [16]. This research field focuses on providing insights by reasoning on the actual process executions, which are recorded in so-called event logs [16]. Event logs group process events in traces, each of which contains the events related to a specific process-instance execution. An event refers to the execution of an activity (e.g., *Apply for a loan*) for a specific process instance (e.g. customer *Mr. Bean*) at a specific moment in time (e.g. on January, 1st, 2018 at 3.30pm).

While process mining has proven to be effective in a wide range of application fields, it has shown its limitation when the process intrinsically allows for a high degree of flexibility [16], or information systems record executions into logs where events are at a lower-level granularity than the concepts that are relevant from a business viewpoint. Both of the problems lead to an "ocean" of observed process behavior. This means that, e.g., if one tries to discover a process model, one obtains a model that is very complex and/or low-level, thus being difficult to interpret.

Figure 1 shows an example of a model that was discovered from an event log that records huge behavioral variabilities. The model was obtained through the new Heuristic Miner [10] and refers to the page-visit behaviour of the *www.werk.nl* web site (see also Section 3.1). Similar models would be discovered using other miners.

Models such as that in Figure 1 are extremely difficult to interpret. They certainly contrast the initial purpose of process mining: conveying interpretable insights and knowledge to process stakeholders and owners.

Similarly to existing related work (see Section 4), here *we advocate the need of abstracting low-level events to high-level events.* However, differently from existing related work, we do not want to rely on the provision of an extensive amount of domain knowledge as many existing approaches require: this can be hard in several domains. On the other hand, we want to avoid completely unsupervised approaches, which naturally show lower accuracy and/or rely on strong assumptions. *Here and in the remainder of the paper, the abstraction of low-level events to high-level events is meant to indicate that the events referring to low-level activities are replaced by other events for high-level activities.*

To balance accuracy and practical feasibility, we aim at a technique that requires process analysts to only feed in knowledge that is limited in quantity and easy to provide. In a nutshell, the idea is that events of the same trace can be clustered into batch sessions (hereafter shorted as sessions) such that the time distance between

SAC '20, March 30-April 3, 2020, Brno, Czech Republic

ACM ISBN 978-1-4503-6866-7/20/03…$15.00
https://doi.org/10.1145/3341105.3373861

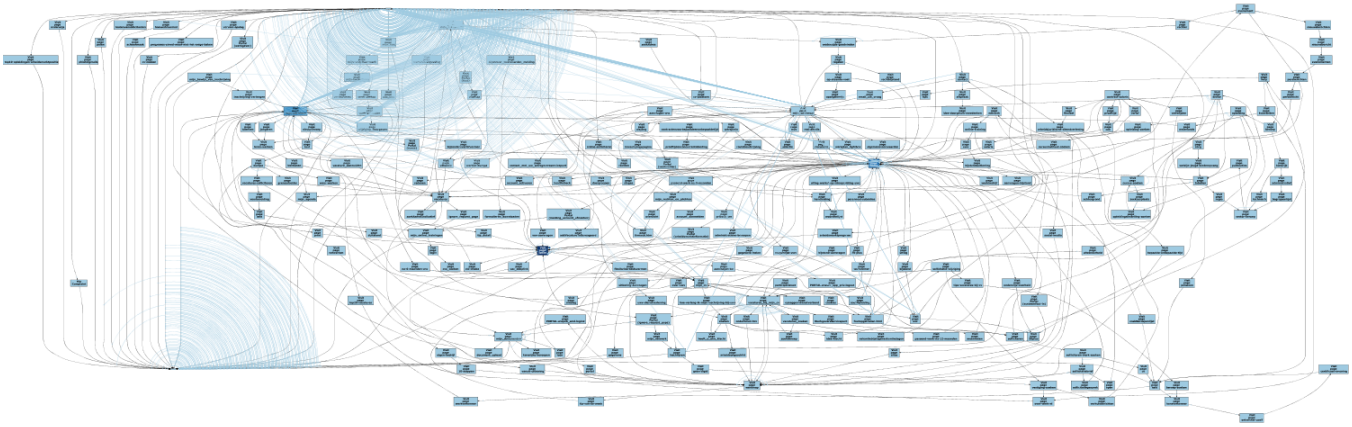


Figure 1: The model for the very flexible process of the users' interactions with the www.werk.nl. The presence of an ocean of diverse behavior generates a model useless to gain insights

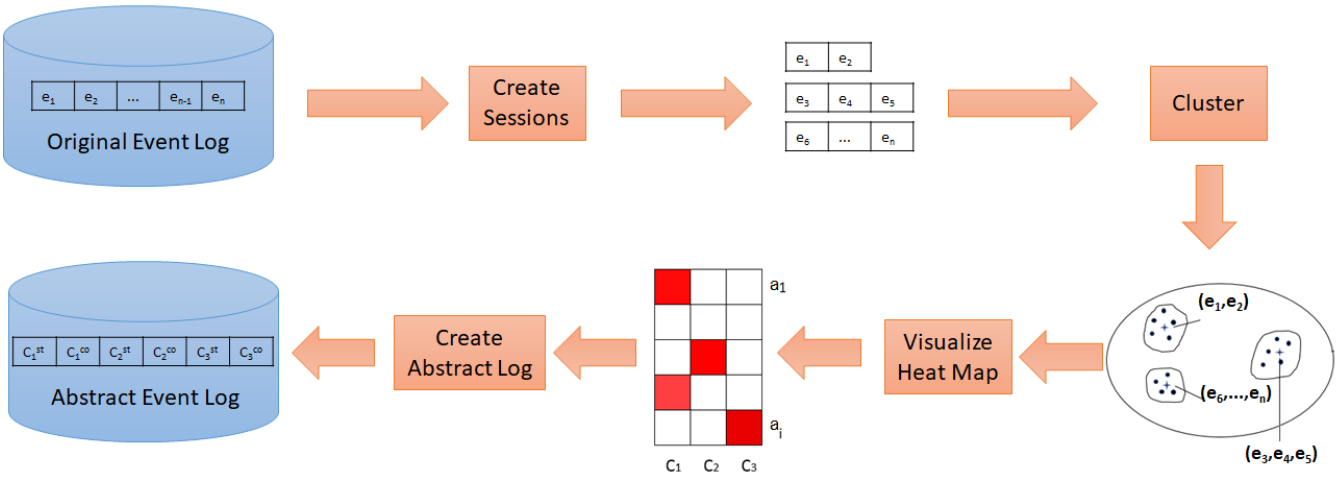


Figure 2: The steps of the abstraction technique based on batch sessions. The technique clusters the sessions with low-level activities $\{a_1, \ldots, a_i\}$, namely the activities associated to the low-level events, into a number of clusters, here $\{C_1, C_2, C_3\}$.

the last event of a session and the first event of the subsequent session is larger than a user-defined threshold. Each trace is seen as a sequence of sessions of events. These sessions are encoded into data points to be clustered; this way, each session is assigned to one cluster. The abstract event log is created such that the entire session is replaced by two high-level events associated with a high-level activity of the same name as the cluster to which the session belongs. The two events per session indicate the start and completion of the session. So, high-level events needs to be named: The centroids of the clusters provide meaningful information for a process stakeholder to identify the activity name of the high-level event that corresponds to each cluster. To support stakeholder in this identification, visualization techniques are foreseen, based on heat maps. However, the latter is optional: e.g., without domain knowledge, each cluster may be named as the concatenation of the names of the low-level activities that clearly stand out in the cluster.

The benefits and feasibility of the proposed technique were assessed on two real-life case studies. The first refers to the www.werk.nl web site. Results show that overcomplex, low-level process models can be converted into high-level counterparts that are accurate according to the process-mining metrics, and that are simply enough to be able to convey information that has business value. The paper reports on a second case study about the management of building-permit requests, which confirms the findings.

The problem raises when events are too-low-level, which is also relatively common in, e.g., health care, customer-journey analysis, home automation, and IoT systems. The event-log abstraction technique is beneficial in these and other domains. In general, one can apply the proposed technique to any domain in which events happen in batches/sessions.

Section 2 introduces the abstraction technique, while Section 3 reports on the evaluation on the two cases. Section 4 compares with

the related work while Section 5 concludes the paper, delineating the avenues of future work.

2 SESSION-BASED EVENT-LOG ABSTRACTION TECHNIQUE

This section introduces our technique to abstract low-level events. The starting point is an **event log**, which consists of a set of traces, where each trace σ is a sequence of events: $\sigma = \langle e_1, \ldots, e_n \rangle$. Each trace groups all the events that refer to executions of activities within the same instance of process (the same user, customer, patients, etc.). Each event e_i carries information. For the scope of this paper, any e_i is at least associated with the activity $\lambda_A(e_i)$ to which the event refers, and the timestamp $\lambda_T(e_i)$ when the event occurred.

Our technique consists of four main steps (see Figure 2). All the traces of the event log are split into sessions; each session of a log trace is a the smallest sub-sequences of the trace such that, for each session, the difference between the timestamp of the last event of the session and that of the first event of the session that follows is larger than a user-defined threshold. Sessions are then extracted and decontextualized from the traces to which they belong. As second step, the sessions are converted into vectors that abstract the behavior observed in the sessions. These vectors are clustered, and hence sessions are also clustered. The third step focuses on visualizing the centroids of the clusters on a heatmap, thus providing information about the most predominant activities for low-level events. This information is used by process analysts to assign activity names to high-level events. The fourth step creates the abstract event log: each session is replaced by two events (e.g. C_1^{st} and C_1^{co} in figure) of the same name as that given to the cluster to which the session belongs. The two events refer to the starting and the completion of the session and, respectively, take on the timestamps of the first and the last event of the session.

2.1 Creation of Sessions

The first step of the technique is to identify the sessions. We introduce a **session threshold** Δ, a time range. For each trace $\sigma = \langle e_1, \ldots, e_n \rangle$ in an event log, we iterate over its events and create a sequence of sessions $\textcircled{s}_\Delta(\sigma) = \langle s_1, \ldots, s_m \rangle$. We create a session $s_k = \langle e_i, \ldots, e_j \rangle$, subsequence of σ, if ***(1)*** the timestamp's difference between e_i and e_{i-1} and e_j and e_{j+1} is larger than or equal to Δ and ***(2)*** the timestamp's difference between two consecutive events in $\langle e_i, \ldots, e_j \rangle$ is smaller than Δ. Also, the concatenation of the events in the sessions of $\textcircled{s}_\Delta(\sigma)$ must result in σ. The following example clarifies:

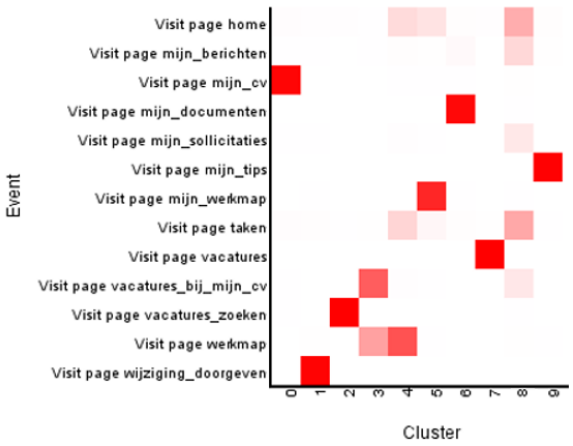
Example 2.1. Consider a trace $\sigma = \langle a_1, b_3, c_4, a_{10}, d_{13} \rangle$. The letter indicates the activity name associated with the event, and the subscript is the timestamp of the event's occurrence (e.g. d occurred at time 13). Assume that the time interval $\Delta = 5$. One can easily see that the time difference between the second occurrence of a and the first of e is greater than the given time interval Δ ($\lambda_T(a_{10}) - \lambda_T(c_4) = 6 > \Delta = 5$), thus resulting in two sessions: $\textcircled{s}_\Delta(\sigma) = \langle s_1, s_2 \rangle$ where $s_1 = \langle a_1, b_3, c_4 \rangle$ and $s_2 = \langle a_{10}, d_{13} \rangle$. Note that their concatenation results in σ.

2.2 Clustering of Sessions

Once the sessions are identified and created, they can be clustered. To this aim, we leverage on off-the-shelf clustering techniques, which require sessions to be encoded as vectors to be subsequently clustered. Several encodings are naturally possible. In the remainder, we introduce two of them, which are of general applicability, available in our implementation, and are used in the case studies reported in this paper (cf. Section 3). Both encodings generate vectors with one dimension for each activity associated with low-level events. In the remainder, the set of activities for low-level events in the log is denoted as $\{x_1, \ldots, x_n\}$, namely, for each $i \in [1, n]$ there exists an event e in some log trace, s.t. $\lambda_A(e) = x_i$.

Frequency-based encoding. Each session $s = \langle e_1, \ldots, e_m \rangle$ is encoded a vector $(v_{x_1}, \ldots, v_{x_n})$ where dimension v_{x_k} is the number of events for activity x_k in s: $v_{x_k} = |e \in s : \lambda_A(e_i) = x_k|$. For instance, sessions s_1 and s_2 of Example 2.1 are encoded as quadruples where the value for the first to the fourth dimension is the number of occurrences of respectively a, b, c, d: namely, the encoding of s_1 and s_2 is $(1, 1, 1, 0)$ and $(1, 0, 0, 1)$, respectively. This encoding is useful when one wants to cluster on the basis of the frequency of occurrence of activities in sessions. Consider, for instance, an online retail shop where each log trace contains one event for each item of product that is added to the basket. Each web-site visit corresponds to a session. The frequency-based encoding makes a vector out of each session with as many dimensions as the products that can be potentially added to a basket: the value of a certain dimension coincides with the quantity bought of the product associated with that dimension.

Duration-based encoding. Each session $s = \langle e_1, \ldots, e_m \rangle$ is encoded a vector $(v_{x_1}, \ldots, v_{x_n})$ where the value of dimension v_{x_k} is the average duration of instances of activity x_k in the session s (a zero value is given if x_k does not occur in s). An accurate measurement of the duration of one instance i of an activity x_k requires the respective session s to include two events $e', e'' \in s$ for the start and completion of i. This way, the duration of i is $\lambda_T(e'') - \lambda_T(e')$. Typical process-mining literature assumes that e' and e'' can be reliable related to some instance of x_k, but it does not assume that one can relate to the same instance. Moreover, often only one event is recorded for each instance of any x_k, i.e. either the start or the completion of the instance. If the event log do not record both start and completion or one cannot reliably associate them to the same instance, the duration of an activity instance can only be estimated as follows. Given a session $s = \langle e_1, \ldots, e_m \rangle$, the duration of the instance associated with any e_i, with $0 < i < m$, is equal to $\lambda_T(e_{i+1}) - \lambda_T(e_i)$, i.e. the timestamp difference of the event that follows e_i and that of e_i. For the last event e_m, we do not have a following event: the duration is assumed to be the average of those instances when it was possible to estimate . To further clarify, let us again consider session s_1 of Example 2.1. We can estimated that the duration of activity a is 2, because a occurred at time 1 and b at time 3. Similarly, b lasted 1 time unit because b was at time 3 and the subsequent event, for c, was at time 4; in the example, the duration of c cannot be estimated because it is the last of the session. In that case, it is assumed to have a duration equal to the average duration of c in all those cases in which the estimated duration was



(a)

Cluster	Name
0	Visit page mijn_cv
1	Visit page wijzigin_doorgeven
2	Visit page vacatures_zoeken
3	Visit page vacatures_bij_mijn_cv
4	Visit page werkmap
5	Visit page mijn_werkmap
6	Visit mijn_documenten
7	Visit page vacatures
8	Visit page taken+home
9	Visit page mijn_tips

(b)

Figure 3: An example of heat map of the cluster centroids (part a) and of names that can be given to the clusters (part b)

possible to be computed, namely when c was not the last event of the session. If we only have the events for the completion of activities, a similar estimation can be made.

Space limitations prevent us to discuss the case when we only have the event's timestamp when activity concludes; however, the estimation is calculated similarly to the situation discussed above when the timestamp is known when activity instances were started.

2.3 Visualization of Heat Maps and Creation of Abstract Event Logs

The results of the second step in Section 2.2 produces a set of clusters $M = \{M_1, \ldots, M_m\}$ of sessions. In a nutshell, given a trace σ with a session's sequence $\langle s_1, \ldots, s_n \rangle$, σ is abstracted as a sequence of high-level events $\langle C_1^{st}, C_1^{co}, \ldots, C_n^{st}, C_n^{co} \rangle$ with the activity name of any high-level event C_i^{st} and C_i^{co} equal to the name assigned to the cluster M_j to which s_i belongs.

Therefore, each cluster $M_j \in M$ needs to be given a name $\textsc{Name}(M_j)$. Process-owner knowledge can be put forward to assign meaningful names. Here, we advocate the use of visual-analytics technique. In particular, we propose the use of heatmaps to visualize the cluster centroids that provide information on how activities for low-level events in sessions are clustered together. An example is in Figure 3(a), which refers to the application to the werk.nl web-site. Each row and column respectively refer to a different activity of different low-level events (i.e. dimension of the clustering space) and to a different cluster.

In particular, the centroid of each cluster is normalized between 0 and 1 and discussed below. Centroids are shown on the heat maps through different red-color intensities, with 0 being white and 1 being the most intense red (or darker color if printed black-on-white). The color for a column X and row Y is proportional to the value of the dimension for low-level activity Y, i.e. Y is the activity associated with the low-level event, in the centroid of cluster X. The normalization of a given centroid $(c_1, \ldots, c_n)$ is achieved by dividing by the sum of the centroid's values: $(\frac{c_1}{sum}, \ldots, \frac{c_n}{sum})$ where $sum = c_1 + \ldots + c_n$. The following example well explains:

Algorithm 1: Creation of an Abstract Event Log

Input: Event Log $\mathcal{L} \in \mathcal{E}^*$, a set $M = \{M_1, \ldots, M_n\}$ of clusters with names $\textsc{Name}(M_1), \ldots, \textsc{Name}(M_n)$
Result: Abstract Event Log

1 $\mathcal{L}' \leftarrow \emptyset$
2 **foreach** $\sigma \in \mathcal{L}$ **do**
3 $\sigma' \leftarrow \langle\rangle$
4 **foreach session** $s = \langle e_1, \ldots, e_m \rangle \in \text{Ⓢ}(\sigma)$ **do**
5 $c \leftarrow \textsc{Encode}(\langle e_1, \ldots, e_m \rangle)$
6 **Pick** $M_i \in M$ **s.t.** $c \in M_i$
7 **Create Events** C^{st} and C^{co} **s.t.**
8 $\lambda_A(C^{st}) = \lambda_A(C^{co}) = \textsc{Name}(M_i)$
9 $\lambda_T(C^{st}) = \lambda_T(e_1)$
10 $\lambda_T(C^{co}) = \lambda_T(e_m)$
11 $\sigma' \leftarrow \sigma' \oplus \langle C^{st}, C^{co} \rangle$
12 **end**
13 $\mathcal{L}' \leftarrow \mathcal{L}' \cup \{\sigma'\}$
14 **end**
15 **return** $(\mathcal{L}')$

Example 2.2. Let us assume the following centroids: $(1, 0, 1, 1, 0, 1)$, $(40, 0, 2, 0, 0, 0)$, $(0, 0, 0, 10, 0, 1)$, $(1, 2, 0, 0, 0, 0)$, $(0, 0, 2, 2, 2, 1)$. The normalization produces $(\frac{1}{4}, 0, \frac{1}{4}, \frac{1}{4}, 0, \frac{1}{4})$, $(\frac{40}{42}, 0, \frac{2}{42}, 0, 0, 0)$, $(0, 0, 0, \frac{10}{11}, 0, \frac{1}{11})$, $(\frac{1}{3}, \frac{2}{3}, 0, 0, 0, 0)$, $(0, 0, \frac{2}{7}, \frac{2}{7}, \frac{2}{7}, \frac{1}{7})$.

Note that we do not normalize by simply dividing by the largest value, such as 42 in Example 2.2. If we did so in Example 2.2, the first, fourth and fifth centroids would be normalized to a vector with almost zero values for all dimensions.

If one obtains such a heatmap as that in Figure 3(a), the stakeholder is largely facilitated to assign names to clusters because almost each cluster is characterized by a centroid with predominant values for one or two dimensions, each associated to a different low-level activity. Algorithm 1 concretely illustrates the procedure. For each trace σ of the log, we firstly create an empty trace σ', which will be eventually added to the abstract event log (line 13). Then, each session of $\text{Ⓢ}(\sigma)$ is encoded again as a vector c (line 4) to determine the cluster M_i to which c and, thus, s belongs (line 5). We create two events C^{st} and C^{co}, which respectively abstract the start and the end of session s. These two events refer to the high-level activity $\textsc{Name}(M_i)$ (i.e., the activity namew to assign to the high-level events): $\lambda_A(C^{st}) = \lambda_A(C^{co}) = \textsc{Name}(M_i)$ (line 8), with the timestamp of C^{st} and C^{co} being respectively equal to that of first and last event of session s (lines 9 and 10). These two events C^{st} and C^{co} are then added to σ' (line 11).

3 EVALUATION

The abstraction technique introduced in this paper has been implemented as a Java plug-in named *Session-based Log Abstraction* in the *TimeBasedAbstraction* package of the nightly-build version of ProM.[1] The implementation features the DBSCAN and K-Means for clustering, and leverages on the ELKI library for DBSCAN [13] and the Weka library for K-Means [20].[2]

The remainder of this section illustrates the application to two case studies: the werk.nl website (Section 3.1), and a building-permit request process (Section 3.2). The two case studies illustrate that it is applicable to more "traditional" process domains (the latter) as well as to processes that drive the interaction between humans and user interfaces of, e.g., web sites.

3.1 Analysis of the werk.nl website

The *werk.nl* web site is run by UWV, which is the social security institute that implements employee insurances and provide labour market services to residents in the Netherlands. Specifically, the web site supports unemployed Netherlands' residents in the process of job reintegration. Once logged in the web site, people can upload their own CVs, search for suitable jobs and, more in general, interact with UWV via messages, and also they can ask questions, file complaints, etc. The www.werk.nl web site is structured into sections of pages and logged-in users can arbitrary switch from any page to anyone else.

The analysis of the web site is motivated by the fact that there is interest to introduce some supporting wizards to improve the customer interaction experience. The starting pointing for designing such wizards is to gain insights into the typical behavior in which the web site is actually used, so that this typical behavior can be converted into a series of wizards.

The analysis was based on an event log composed by 335655 events, divided in 2624 traces: it records the behavior of the logged-in visitors in the period from July, 2015 to February, 2016.[3]

Figure 1 has shown that the model constructed without abstraction is overly complex and, thus, is not helpful in constructing those wizards. This justifies the relevance of applying the event-log abstraction technique discussed in Section 2. For this case, we used the duration-based encoding (cf. Section 2.2) to cluster the web-site interaction sessions. In this case, the duration-based encoding gives higher importance to web pages where visitors remain longer. Conversely, a frequency-based encoding would have given, e.g., higher weights to three web-page visits of one minute each instead of one 30-minute-long visit: the one-minute-long visit could be linked, e.g., to visitors that mistakenly click on web link and visit the target page. The session threshold Δ was set to 15 minutes, because it coincides with the timeout of www.werk.nl.

Once the sessions are encoded, we clustered the encoded vectors, using DBSCAN algorithm. The generation of the clusters via DBSCAN took nearly 2 hours on a low-profile laptop with 8 Gb of RAM. The clusters' centroids were visualized through the heatmap

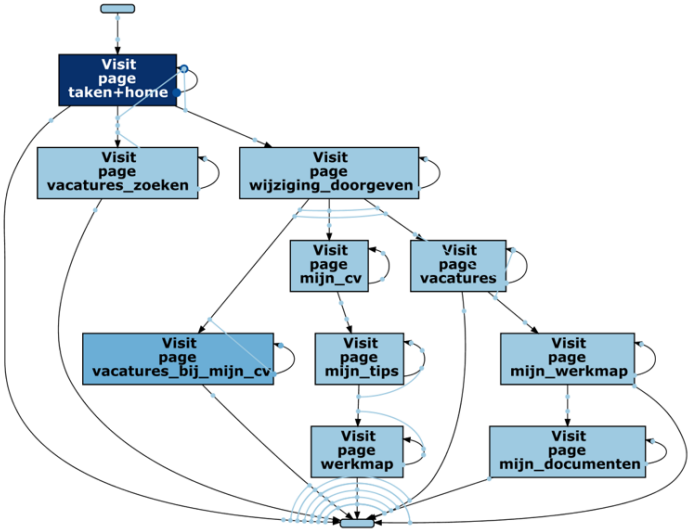


Figure 4: Process model produced by the Heuristic Miner [10] on 70% of the abstract event log of the werk.nl dataset, clustering via DBSCAN.

in Figure 3(a). To help stakeholders, the plug-in removes the rows referring to low-level activities that, when normalized, are associated with nearly-zero values of all dimensions.

Figure 3(a) shows very interesting clustering results: With the exception of cluster 8, the centroids illustrate that the clusters are characterized 1-3 pages that are visited for long periods. In fact, five out of 10 clusters (50%) refer to web-site sessions that are centered around one page, only. Without additional domain knowledge, each cluster could only be named after the web-page name, activity of the corresponding low-level event, associated with the most intense color in the heat map, namely the dimension with the largest value for the centroid of the cluster. The clusters are hence given names as in Table 3(b).

Once names are assigned to clusters, we generated an abstract event log, according to our technique. The quality of the abstract event log to generate an accurate model was assessed by randomly splitting the log into a 70% part, used for model discovery, and a 30%, for testing. The DBScan algorithm naturally computes outliers, namely points that are not assigned to any cluster. Results show that, if those outliers are simply filtered out, the quality of the discovered model is significantly dropped (see discussion below, summarized in Table 1). Therefore, we performed a post-processing step: each session assigned to no cluster is manually inserted into the closest cluster. For each session s encoded as a vector v, the closest cluster is that associated with the centroid c at the minimum distance from v, considered among all the centroids of the clusters.

The abstract event log with the manual assignment of outliers was used as input for the new Heuristic Miner [10], thus discovering the model in Figure 4, using the Causal-Net notation [16].

The same procedure was employed to discover a high-level model with K-Means, using the same 70% of the traces for discovery, and the same temporal threshold and encoding as for DBSCAN. Space

[1] ProM Web-site: http://www.promtools.org

[2] The choice for the ELKI library for DBSCAN was motivated on the fact that the ELKI library showed to perform much faster than Weka.

[3] The dataset is available at https://doi.org/10.4121/uuid:01345ac4-7d1d-426e-92b8-24933a079412.

	K-Means	DBSCAN Post-Processing	DBSCAN No Post-Processing
Fitness	0.6637	0.6270	0.2785
Precision	0.33192	0.74779	0.68247
Generalization	0.99962	0.99996	0.99998
Simplicity	81	91	79

Table 1: Measures of the quality of the models discovered on the log abstracted through K-Means and DBSCAN. For the DBSCAN, we report on the values when the postprocessing to manually insert outlier was and was not performed.

limitations prevent us from showing the model discovered via K-Means, which is however available in the technical report that extends this submission [2]. Note that, compared with DBSCAN, K-Means requires one to explicitly set the number of clusters to create. We feature the *Elbow Method* to determine a good number of clusters to create [7]. This good number should balance the number of clusters versus the error within the clusters, where the error is defined as the average distances of the points of a cluster from the respective centroid. After employing the Elbow Method in our setting, we opted to create ten clusters.

The quality of these models was assessed through the classical process-mining metrics of fitness, precision, generalization and simplicity [16]. Fitness was computed on the 30% of abstract log that was not used for discovery. This is accordant with typical machine-learning methods of verifying "recall" on a set of instances that were not used for learning the model. In our setting, this set of instances corresponds to the set of traces, i.e. a sub log, that was not used for discovering the model. Conversely, precision and generalization were computed on the entire abstract log. Finally, simplicity was measured as the sum of activities, arcs and bindings of the causal nets. Since fitness, precision and generalization are traditionally defined on Petri nets [16], causal nets were converted to Petri nets using the implementation in [10]. The resulting Petri nets were manually adjusted to ensure soundness while not adding extra behavior. Of course, *to keep the comparison fair, all models were discovered by the Heuristic Miner [10], using the same configuration of parameters*, which was also the configuration used to discovered the model in Figure 1, i.e. without abstracting the event log.

Table 1 illustrates the results of the comparison of the models discovered through the abstract event logs obtained via K-Means and DBSCAN. All models nearly generalize equally well. The abstract model when applying DBSCAN without post processing shows very poor fitness, which is conversely satisfactory when applying K-Means or DBSCAN with post processing. Focusing on precision, the model of DBSCAN with post-processing is characterized by a precision that is 2.25 times than the precision of the K-Means model. The model obtained via DBSCAN with post processing is just slightly more complex than the others, ca. 15%. As a conclusion, DBScan with post-processing has produced a better model, in terms of mediating fitness, simplicity, precision and generalization. Intuitively, this is not surprising: DBScan is based on maximizing the cluster density, maximizing the similarity of sessions within the same cluster.

In conclusion, the model in Figure 4 is the most preferable and unarguably more understandable, if compared with the non-abstract model in Figure 1. From a business viewpoint, it illustrates that typical users navigate the werk.nl web site as follows. During the first session, users visit the home page and, also, page *taken* (Dutch for *tasks*), where they can see the tasks assigned by UWV (e.g. to upload certain documents). If no tasks are assigned to do via the web site, the interaction with the web site completes. If any tasks are, users look for jobs to apply for (page *vacatures_zoeken*) and/or amend the information that they previously provided (page *wijziging_doorgeven*). If information is amended, usually an updated curriculum is uploaded (cf. the branch of the model starting with page *mijn_cv*) and/or the visitor looks and possibly applies for jobs (cf. the branches of the model starting with pages *vacature* and *vacature_bij_mijn_cv*, which are either both executed or both skipped). Looking at statistics, the mean and median duration of the web-site interaction (i.e. the log traces) is around 20 weeks (more than 4 months) and, hence, the visiting sessions are certainly temporarily spread. One can also observe that every session type is usually repeated multiple times, and this is likely due to the fact that the corresponding tasks are carried on through similar sessions in consecutive days. It is, however, remarkable that the model does not contain larger loops involving different session types. This means that the web site is visited in conceptual sections: when users start access pages of a given section, the pages of previous sections will no longer be visited. Note that the web site does not define sections, nor does it restrict the order with which pages can be visited. In fact, this testifies the benefits of introducing wizards. We acknowledge that information is lost in the abstraction. However, this loss is justified by gaining comprehensible business knowledge. As a matter of fact, this model was shown to one UWV's stakeholder, who clearly indicated that the results show the most understandable analysis of the web-site behavior that I have seen, certainly beyond the results seen in previous analyses, including those of the 2016 BPI challenge.[4]

3.2 Evaluation on a Building-Permit Process

This second case study refers to the execution of process to manage building-permit applications in a Dutch municipality.[5] There are 304 different activities denoted by their respective English name as recorded in attribute *taskNameEN*. The event log spans over a period of approximately four years and consists of 44354 events divided in 832 cases. Figure 5 shows the model discovered with the *Inductive Miner - Infrequent Behavior* [9], using the default configuration.[6] The model exactly shows the same problems as that in Figure 1: The large variability has made the miner discover an overly complex model. See, e.g., the large OR split around the area highlighted by a circle in the picture. We applied the abstraction technique to the event log, using the frequency-based encoding (cf. Section 2.2) and the DB-SCAN clustering algorithm with post processing, which proved to perform better for the first case study reported in Section 3.1. A session threshold of 8 hours was employed so that the events of the same day were put in the same (work) session.

[4] BPI challenge 2016 - https://www.win.tue.nl/bpi/doku.php?id=2016:challenge

[5] The event log is available at http://dx.doi.org/10.4121/uuid:63a8435a-077d-4ece-97cd-2c76d394d99c

[6] We acknowledge that the model in Figure 5 is not readable. However, the purpose is not to read the model's details, but to gain an impressions of its lack of a structure that can convey information

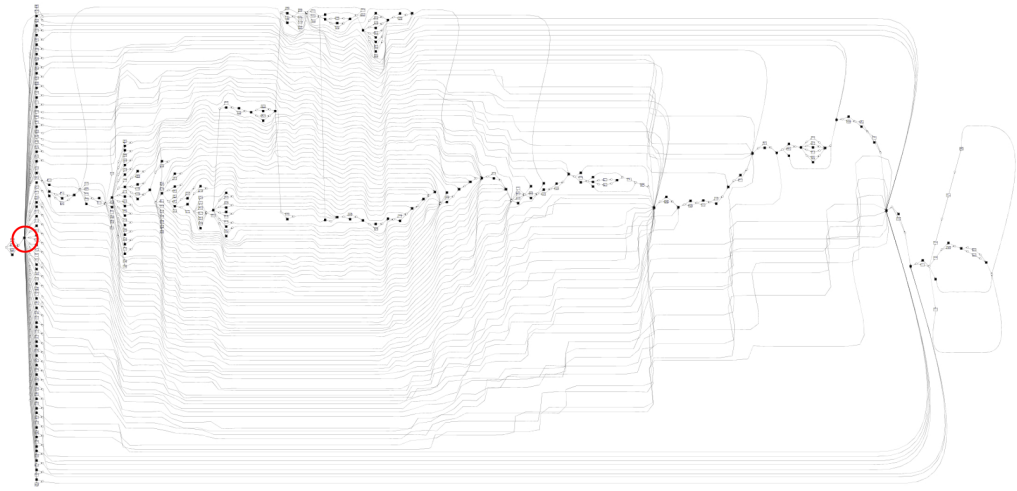
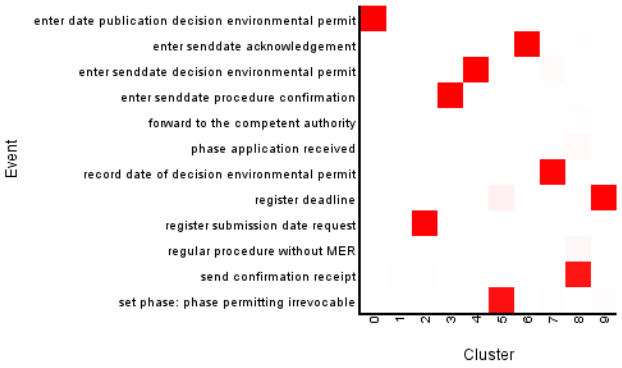


Figure 5: Building-permit process model produced by the Inductive Miner without abstraction: overly complex to be insightful.

The heatmap is shown in Figure 6(a), which is structurally similar to Figure 3(a) where each cluster centroid has significantly non-zero values for the dimensions referring to one or two low-level activities. Similarly to the first case study, clusters were given the same name as the low-level activity with a significantly intense colour in the heat map. If more than one low-level activity is associated with an intense colour in the map, we concatenated the names of these activities. The names assigned to clusters are shown in Figure 6(b).

The abstract event log was then generated and used as input for the *Inductive Miner - Infrequent* with default parameter values, namely the same as for the not-abstracted model in Figure 5. This yielded the model in Figure 7, which is unarguably simplified, emphasising the most salient behavioral aspects. This model is a good representation of the actual behavior: its fitness is 0.79. Unfortunately, it was not possible to compute precision and generalization because the reference ProM implementation [16] never terminated the computation.



(a)

Cluster	Name
0	enter date publication decision environmental permit
1	completed subcases content
2	register submission date request
3	enter senddate procedure confirmation
4	enter senddate decision environmental permit
5	set phase: phase permitting irrevocable & register deadline
6	enter senddate acknowledgement
7	record date of decision environmental permit
8	forward to the compotent authority & send confirmation receipt & regular procedure without MER & phase application received
9	register deadline

(b)

Figure 6: The heat map of the cluster centroids for the building-permit process (part a) and the names given to the clusters (part b)

3.3 Final Remarks

We employed different process-discovery algorithms for the two case studies, namely ILP Miner, Heuristic Miner, Inductive Miner, and Alpha+ Miner [16]. For the sake of space, we only report on the results for the discovery algorithm that worked better in each case study. In fact, different algorithms rely on different assumptions, and hence the quality of the results depend on the nature of the specific process being analyzed.

For each case study, the paper only reports models that were mined using the same discovery algorithm, thus keeping the comparison fair. Very importantly, it never happened that, without abstracting the event log, one discovery algorithm was able to mine a process model that was simple and conveying information. Therefore, the hypothesis remains valid that event-log abstraction is necessary in certain domains to mine process models that provide useful, actionable insights.

4 RELATED WORK

A large body of research has been conducted on log abstraction. It can be grouped in two categories: supervised and unsupervised abstraction. The difference is that supervised abstraction techniques

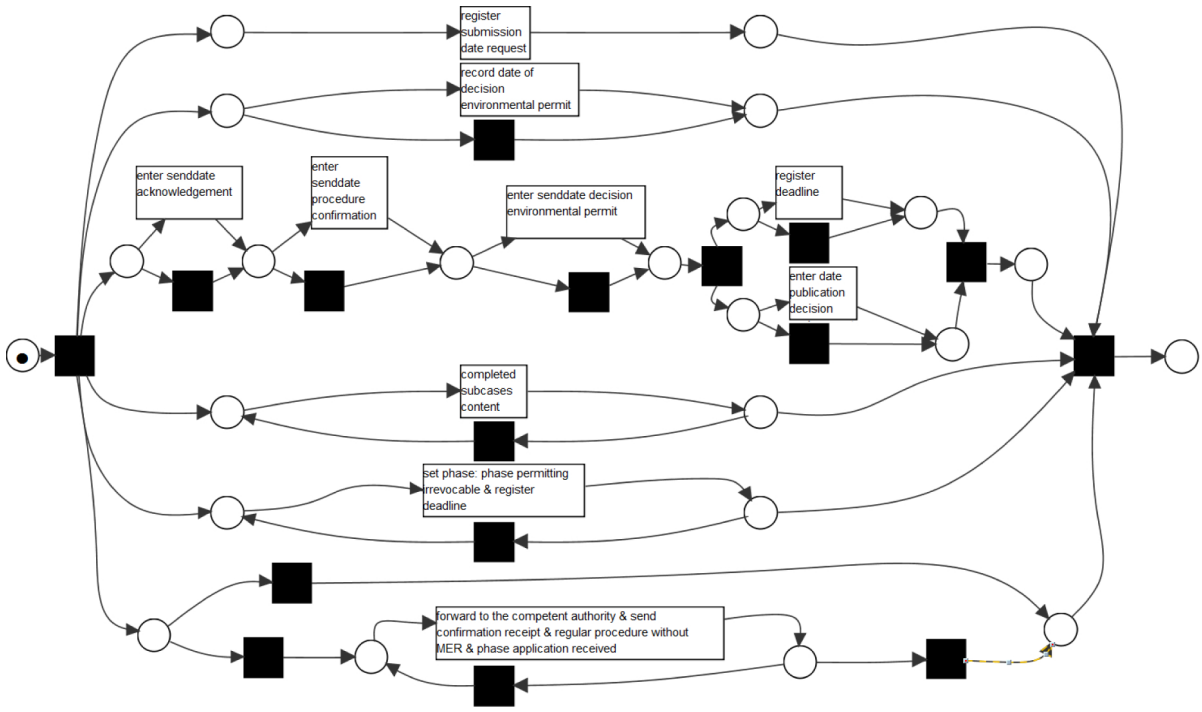


Figure 7: Building-permit process model produced by the Inductive Miner with abstraction, clustering via DBSCAN.

require process analysts to provide domain knowledge, while unsupervised does not rely on additional information. Below, we illustrates how, on the one hand, supervised methods often require vast domain knowledge (e.g. through process models, Markov chains or mapping ontologies), which is not always possible to provide. On the other hand, unsupervised methods show limitations, related to the absence of any external knowledge.

Supervised Abstraction Methods. Baier et al. provide a number of approaches that, based on some process documentation, map events to higher-level activities [1], using log-replay techniques and solving constraint-satisfaction problems. The idea of replaying logs onto partial models is also in [11]: the input is a set of models of the life cycles of the high-level events, where each life-cycle step is manually mapped to low-level events. Ferreira et al. [5] rely on the provision of one Markov model, where each Markov-model transition is a different high-level activity. In turn, each transition is broken down into a new Markov model where low-level events are modelled. Fazzinga et al. [4] assume process analysts to provide a probabilistic process model with the high-level events, along with a probabilistic mapping between low-level events and high-level events. It returns an enumeration of all potential interpretations of each log traces in terms of high-level events, ranked by the respective likelihood. Montani et al. [12] propose an abstraction technique that requires one to provide a complex ontology based on a hierarchy of concepts. In [15], authors propose a supervised abstraction technique that is applicable in those case in which annotations with the high-level interpretations of the low-level events are available for a subset of traces.

Unsupervised Abstraction Methods. Log abstraction is related with episode mining and its application to Process Mining (a.k.a. discovery of local process models) [8, 14]). In fact, Mannhardt and Tax propose a method that combines local process model discovery with the supervised abstraction technique in [11]. However, the technique relies on the ability to discover a limited number of local process models that are accurate and cover most of the low-level event activities. Günther et al. cluster [6] events looking at their correlation, which is based on the vicinity of occurrences of events for the same low-level activity in the entire log . Clustering is also the basic idea of [17] to cluster events through a fuzzy k-medoids algorithm. Both [17] and [6] share the drawback that the time aspects are not considered and, thus, they can cluster events that are temporarily distant (e.g. web-site visits that are weeks far from each other). Also, [17] only aims to discover a fuzzy high-level model, instead of abstracting event logs to enable a broader process-mining application, whereas [6] assumes a transitive nature of the property of activity correlation, which does not always hold. See Figure 3(a): cluster 3 shows a correlation between *Visit page werkmap* and *Visit page vacature_bij_mijn_CV* and cluster 4 shows a correlation between *Visit page werkmap* and *Visit page taken*, while no correlation exists between *Visit page vacature_bij_mijn_CV* and *Visit page taken*. Finally, van Eck et al. [18] illustrate a technique to gather observations from sensor data, encode and cluster them in a similar way as our approach does. However, they assume that events (in fact, sensor observations) are generated at a constant rate.

Note that the problem of tackled in this paper of abstracting event logs is different from the problem of event-log trace clustering.

Event-log clustering aims to split the event log into sub logs that show homogenous behavior, but the single traces are not altered or abstracted [3]. The vast literature on event-log clustering only shares one minor aspect with the technique reported in this paper: the use of clustering techniques on events.

The topic of this paper is also linked to the field of research of studying the logs of interaction of human with user interfaces (UIs) of software, machines, or of the elements of IoT systems (see, e.g., the results of the LIVVIL workshop [19]). However, the study of the logs of human-UI interaction is a field of research that only targets one type of event logs. This allows for having techniques with effective, precise results, however at the cost of being only applicable in that context. Conversely, the technique reported in this paper is of more general applicability, as illustrated in case study with the building-permit process.

5 CONCLUSION

Abstracting and grouping low-level events to high-level events is a problem that is receiving a lot of attention. Often, event logs are not immediately ready to be used because they model concepts that are not at the right business level and/or they exhibit a too broad variety of behavior to be summarized into one sufficiently simple model. We report on a technique where very limited domain knowledge is necessary, trying to balance the limitations of unsupervised techniques and the request of a vast knowledge of those supervised (cf. Section 4).

The idea behind our technique is that a trace can be regarded as a sequence of batch sessions, which are the shortest sequences of low-level events such that the timestamp difference between the first event of a session and the last of the previous is larger than a given user-define threshold. Each session is replaced in the abstract event log by two high-level events marking the start and completion of a session. Sessions are clustered and are given names. The high-level activity associated with each pair high-level events is given the same name as that of the cluster to which the session belongs. Heatmaps are used to help domain experts give meaningful high-level names to clusters and, consequently, to sessions.

Section 3 reports on the successful evaluation of the proposed technique on the werk.nl case study, discussing both a quantitative and a qualitative analysis. Our technique allows generating abstract event logs that are simple and insightful for process owners, because they focus on concepts at a higher level of abstraction. Quantitatively, when used for discovery, the generated event logs enable mining models that can balance the typical process-mining metrics: fitness, precision, generalization and simplicity [16]. A second case study on building-permit application further assess the benefits already observed after the analysis of the werk.nl web-site.

The technique does not depend on any clustering algorithm, and this explains why concrete algorithms are only mentioned in Section 3. While we acknowledge that a more thorough assessment is necessary, Section 3 shows that the best performances are with DB-Scan, which has the advantage of automatically computing the best number of clusters. In sum, our technique requires the provision of little knowledge: considering that the provision of the cluster names is optional (cf. Section 2.3), our technique only requires the session as mandatory input, if clustering is done via DBSCAN.

In the future, we aim to work towards a more accurate clustering, considering the entire event payload, instead of just limiting to the activity names. As an example, the application to the *werk.nl* case study could benefit if one added clustering dimensions related to the customer age, gender, geographic locations, etc., thus providing extra information towards a more accurate clustering. Furthermore, the number of low-level activities is generally large, creating sparse clustering points that can compromise the quality: We aim to reduce this number to consider before clustering.

REFERENCES

[1] Thomas Baier. 2015. *Matching Events and Activities.* PhD dissertation. University of Potsdam.

[2] Massimiliano de Leoni and Safa Dündar. 2019. From Low-Level Events to Activities - A Session-Based Approach (Extended Version). *arXiv.org* abs/1903.03993 (2019). http://arxiv.org/abs/1903.03993

[3] Jochen De Weerdt. 2018. *Trace Clustering.* Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-63962-8_91-1

[4] Bettina Fazzinga, Sergio Flesca, Filippo Furfaro, Elio Masciari, and Luigi Pontieri. 2015. A probabilistic unified framework for event abstraction and process detection from log data. In *Proceedings of the 23th OTM Confederated International Conference on Cooperative Information Systems (LNCS)*, Vol. 9415. Springer, 320–328.

[5] Diogo R Ferreira, Fernando Szimanski, and Célia Ghedini Ralha. 2013. Mining the low-level behaviour of agents in high-level business processes. *International Journal of Business Process Integration and Management 8* 6, 2 (2013), 146–166.

[6] Christian W Günther, Anne Rozinat, and Wil M. P. van der Aalst. 2009. Activity mining by global trace segmentation. In *Proceeding of the 7th International Conference on Business Process Management.* Springer, 128–139.

[7] D. J. Ketchen and C. L Shook. 1996. The application of cluster analysis in strategic management research: An analysis and critique. *Strategic Management Journal* 17, 6 (1996), 441–458.

[8] Maikel Leemans and Wil M. P. van der Aalst. 2015. Discovery of Frequent Episodes in Event Logs. In *The 4th International Symposium on Data-Driven Process Discovery and Analysis, (SIMPDA 2014) (LNBIP)*, Vol. 237. Springer, 1–31.

[9] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. 2013. Discovering Block-structured Process Models From Event Logs - A Constructive Approach. In *Proceedings of the 34th International Conference on Application and Theory of Petri Nets and Concurrency (Petri Net 2013) (LNCS)*, Vol. 7927. Springer, 311–329.

[10] Felix Mannhardt, Massimiliano de Leoni, and Hajo A. Reijers. 2017. Heuristic Mining Revamped: An Interactive Data-aware and Conformance-aware Miner. In *Proceedings of the BPM Demo Track and BPM Dissertation Award at 15th International Conference on Business Process Management*, Vol. 1920. CEUR-WS.org.

[11] Felix Mannhardt, Massimiliano de Leoni, Hajo A. Reijers, Wil M. P. van der Aalst, and Pieter J. Toussaint. 2016. From Low-level Events to Activities - A Pattern-based Approach. In *Proceedings of the 14th International Conference on Business Process Management (LNCS)*, Vol. 9850. Springer, 125–141.

[12] Stefania Montani, Giorgio Leonardi, Manuel Striani, Silvana Quaglini, and Anna Cavallini. 2017. Multi-level abstraction for trace comparison and process discovery. *Expert Systems with Applications* 81 (2017), 398 – 409.

[13] Erich Schubert and Arthur Zimek. 2019. ELKI: A large open-source library for data analysis - ELKI Release 0.7.5 "Heidelberg". *CoRR* abs/1902.03616 (2019). arXiv:1902.03616 http://arxiv.org/abs/1902.03616

[14] Niek Tax, Natalia Sidorova, Reinder Haakma, and Wil M.P. van der Aalst. 2016. Mining Local Process Models. *Journal of Innovation in Digital Ecosystems* 3, 2 (2016), 183 – 196.

[15] Niek Tax, Natalia Sidorova, Reinder Haakma, and Wil M. P. van der Aalst. 2018. Event Abstraction for Process Mining Using Supervised Learning Techniques. In *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016.* Springer, 251–269.

[16] Wil M. P. van der Aalst. 2016. *Process Mining - Data Science in Action.* Springer.

[17] Boudewijn F van Dongen and Arya Adriansyah. 2009. Process mining: fuzzy clustering and performance visualization. In *Proceedings of the 7th International Conference on Business Process Management.* Springer, 158–169.

[18] M. L. van Eck, N. Sidorova, and W. M. P. van der Aalst. 2016. Enabling process mining on sensor data from smart products. In *Proceedings of the Tenth IEEE International Conference on Research Challenges in Information Science (RCIS).*

[19] Romain Vuillemot, Jeremy Boy, Aurélien Tabard, Charles Perin, and Jean-Daniel Fekete (Eds.). 2016. *Proceedings of the workshop LIVVIL: Logging Interactive Visualizations and Visualizing Interaction Logs.* Baltimore, United States. https://hal.inria.fr/hal-01535913

[20] Ian H. Witten, Eibe Frank, and Mark A. Hall. 2011. *Data Mining: Practical Machine Learning Tools and Techniques* (3 ed.). Morgan Kaufmann, Amsterdam. http://www.sciencedirect.com/science/book/9780123748560