The Interplay Between High-Level Problems And The Process Instances That Give Rise To Them*

Bianka Bakullari✉[1][0000−0003−2680−0826], Jules van Thoor[2], Dirk Fahland[2][0000−0002−1993−9363], and Wil van der Aalst[1][0000−0002−0955−6940]

[1] RWTH Aachen University, Germany
[2] Eindhoven University of Technology, Netherlands
{bianka.bakullari, wvdaalst}@pads.rwth-aachen.de
d.fahland@tue.nl

Abstract. Business processes may face a variety of problems due to the number of tasks that need to be handled within short time periods, resources' workload and working patterns, as well as bottlenecks. These problems may arise locally and be short-lived, but as the process is forced to operate outside its standard capacity, the effect on the underlying process instances can be costly. We use the term *high-level behavior* to cover all process behavior which can not be captured in terms of the individual process instances. The natural question arises as to how the characteristics of cases relate to the high-level behavior they give rise to. In this work, we first show how to detect and correlate observations of high-level problems, as well as determine the corresponding (non-)participating cases. Then we show how to assess the connection between any case-level characteristic and any given detected sequence of high-level problems. Applying our method on the event data of a real loan application process revealed which specific combinations of delays, batching and busy resources at which particular parts of the process correlate with an application's duration and chance of a positive outcome.

Keywords: batch · workload · throughput time · outcome

1 Introduction

1.1 Motivation

Process mining techniques analyze event data stored in information systems in order to get insights of real business processes [1]. Organizations strive to improve their running processes by reducing cost and waste, improving resource utilization and customer satisfaction, and so on. Many Key Performance Indicators (KPIs) describe the process in terms of the individual process instances (also called *cases*), e.g., by referring to the average time it takes for a case to complete the process (the *throughput time*), the average accumulated cost or positive outcome rate. Process instances, however, do not run in isolation. From

* We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

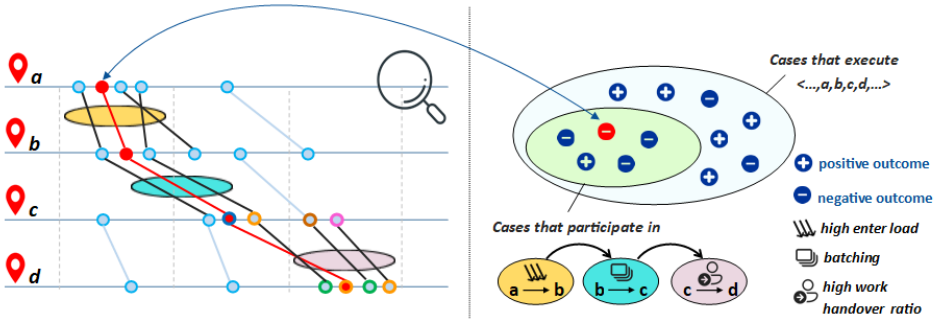


Fig. 1. An illustration of the approach: On the left, a sketch of the performance spectrum [5] showing process instances running through segments (a, b), (b, c) and (c, d). At each segment, several event pairs give rise to various patterns: high entering load at (a, b) (top cloud), batching at (b, c) (middle cloud), and high work handover ratio at (c, d) (bottom cloud). The cases which are involved in this pattern sequence are considered as "participating" w.r.t. that sequence. Given a case-level property, we analyze how its value changes when comparing these cases (e.g., the process outcome of the participating red case is negative) to the cases which visited the same locations where the pattern emerged (here (a, b), (b, c), and (c, d)), but did not give rise to such pattern.

this viewpoint, cases that are simultaneously active in a process resemble cars moving along traffic. Cars can cause traffic jams which, in turn, cause delays and accidents. Similarly, cases may overload the process and the workers, leading to congestion and delays. Moreover, when attending to multiple active cases, resources may execute work in *batches* which, in turn, also influences the manner in which process traffic moves forward. We refer to this kind of emergent process behavior, which is not detectable at the level of the individual instances, as *high-level behavior*. This behavior is *dynamic*; that is, it may arise locally and be short-lived, but it can have an influence on the process runs of the cases active at that time. Nevertheless, similar to traffic jams, there is always a specific set of cases involved whenever such behavior emerges. On one hand, the characteristics of a case may aggravate the emergence of high-level behavior, e.g., a demanding case can block resources for longer time periods. On the other hand, the outcome of a case can also be affected by high-level behavior occurring throughout its run, e.g., a case may not receive the necessary attention if by chance it enters the process in a busy period. There is obviously an interplay between the high-level behavior that arises in the process and the cases which give rise to it. Our method explores this interplay by detecting which patterns of high-level behavior emerge surprisingly often from specific case types. The advantage of possessing this knowledge can be manifold. Depending on the case property at hand, one can adjust the process for specific types of cases in order to avoid undesired but expected high-level problems, or one can make a better online prediction of the progress of a case given its involvement in specific patterns of high-level behavior.

1.2 Example

Suppose that the red lines in Fig. 1 describe the process run of a case which executes activities $\langle a, b, c, d \rangle$. Assume that, in this process, it is unusual to observe four cases entering segment (a, b) simultaneously in a short time frame (the lines within the top yellow cloud). Moreover, multiple cases including the red case execute activities b and c very close to each other with a very similar waiting time in-between (the lines within the middle blue cloud). Later in the process, four cases traverse segment (c, d) in a short time frame (the lines within the bottom violet cloud) and work is handed over from four resources (four different colors encircling the c events) to only two resources (only green and orange encircling the four d events). Assuming that this work handover ratio is unusually high, we can claim that in this example, the red case is involved in three patterns at three different process segments it traverses: *high enter load* at (a, b), *batching* at (b, c) and *high work handover ratio* at (c, d). Now suppose that the red case turns out to have a negative outcome in the process (see the red case's minus sign and positioning in the right side of Fig. 1). The question arises whether participating in this specific sequence of high-level patterns increases or decreases the likelihood of a negative outcome in the process. In this study, we evaluate whether a particular type of cases is disproportionately represented within the case set that generates a specific episode of high-level behavior. When such a situation occurs, we consider the connection between that specific episode and case property as a valuable insight into the behavior of the process.

1.3 Approach

As shown in the previous example, diverse types of high-level patterns can emerge at different locations and times within the process. To be able to compute how strongly cases' participation in high-level behavior correlates with any particular case characteristic, we first need to define what high-level behavior may look like (the clouds in Fig. 1). In this work, we introduce different types of high-level behavior at the segment level that relate to load (enter and exit rates), resource busyness (handover ratio and workload), and working patterns (batches and delays). We use the idea introduced in previous work [2] and conceptualize each outlier observation of such behavior using *high-level events*. It is worth emphasizing that multiple concurrently active cases can give rise to various forms of high-level behavior. This behavior can refer not only to process traffic and workload but also e.g., compliance with regulations which guide how the process should be executed given any particular set of active cases. Within this work, we outline high-level behavior that is specifically related to congestion as it is commonly observed across different process domains. The method could, however, be easily extended to any other type of high-level problem that arises from a set of cases that pass through a process segment in close time proximity. Given a high-level event, we determine what qualifies a case as "(non-)participating" (where to assign each case w.r.t. the sets depicted as circles in the right side of Fig. 1). Any two high-level events with sufficient overlap in time, location and underlying case

sets are assumed to be correlated, leading this way to sequences of subsequently connected high-level events (such as the sequence consisting of the three clouds in Fig. 1). A case participates in such a *high-level path* whenever its events are involved in each high-level event comprising the path. In Fig. 1, these are the cases whose black lines are caught up in all three clouds, such as the red case. In order to investigate the correlation between a case-level characteristic and a case's participation in a given high-level path, we compare the participating cases only to those cases which visit the same locations in the process where the high-level behavior was observed. In the example from Fig. 1, these would be segments (a, b), (b, c), and (c, d) in this order. Hence, in this work, we define the "non-participating" cases to be those case which are not participating, but could have participated from a control-flow perspective.

2 Related Work

Many of the recently developed process mining techniques acknowledge that the progress of cases in a process is influenced by the coexistence of other cases with which they must share process capacities. In [3], the authors aim to improve the rate of positive outcomes in the process. They propose appropriate interventions on changeable case aspects after having identified which treatments have a high causal effect on which case types. This idea is taken further in [4] where the appropriate time for applying a given time reducing intervention on a running case is determined. This decision is based on the causal effect of the intervention which, in turn, includes the number of active cases as an additional feature in the learning process. Results in [15] show that the prediction of case delay at a certain activity is improved when the model is either a transition system whose state space is extended with system load information, or the model is based on queueing theory. The method proposed in [13] shows how information regarding workload and resource availability can be extracted from raw event data and encoded into congestion graphs. From these congestion graphs, congestion-related features can be extracted which are then used for predicting the time until next activity. Inter-case dependencies are also acknowledged in [14] where the current case prediction also factors in the predictions of the cases coexisting with the current one. All these approaches acknowledge that process instances do not run in isolation and as such, integrating dedicated features which capture congestion to train time prediction models improves their accuracy. While many of the high-level patterns we analyze in this work relate to process congestion, we do not encode our observations as additional features describing our running cases for prediction purposes. Instead, each sequence of recurring outlier observations represents an explicit variant of high-level behavior which is a trait of the process itself. We then look back into the "low-level" cases which were part of these observations to reason on whether a specific case type is over- or underrepresented in this group.

The performance spectrum [6] clearly showed that processes—even within the same segment—exhibit non-stationary behavior which is not observable un-

der aggregation. The emerging patterns can reveal batching behavior [8] which can have an influence on performance. In [9], the authors use visual analytics techniques based on the performance spectrum to demonstrate how errors in remaining time prediction are reduced when information on batching behavior is encoded in the learning process. Our method conceptualizes many of the patterns that can be seen in the performance spectrum—including batching—through dedicated events, which can then be mined for further automated analysis.

Several recent methods have been developed which analyze how resources handle tasks from concurrently active cases within a process. In [16], the authors provide insights into how resources prioritize their work by employing specific queueing disciplines when processing individual tasks. In [10], the authors detail the detection of batching behavior not only at the level of individual tasks but also across several linked activities. Moreover, the approach outlined in [11] considers various batch behaviors concerned with multiple perspectives such as activities, resources, and data perspectives as well as allowing for batch detection even when batching is temporarily interrupted. In [18], the authors introduce an enhanced resource profiling technique that considers not only the executed activities, but also the context (duration, case attributes) as well as multitasking. In our work, we incorporate the resource dimension within the high-level events that describe outlier observations concerning batching and workload in specific process segments. However, these observations are local and temporary and their emergence becomes relevant only in relation to the types of cases involved.

In [7], the authors introduce the concept of contextual association, wherein a group of cases exclusively exhibits concept drift whenever a shared object is subject to a change. In our work, cases which give rise to high-level behavior are contextually associated due to their shared location and time in which the behavior is observed. In this scenario, the term “context" solely represents the *coordinate* of a temporary observation in the process.

The idea of conceptualizing outlier behavior related to load and delays as events themselves was first introduced in [17]. In that work, the emerging *system-level events* from a Baggage Handling System (BHS) were connected based on time and place proximity. The resulting cascades revealed how undesired system-level behavior arose and propagated throughout the BHS. Similar work extending this idea was done in [19] where DBSCAN is used to find frequent sequences of anomalies arising at the system-level.

The method we propose in this paper fits in the *high-level event mining* framework we introduced in [2]. Each high-level event consists of the type of behavior detected, the entity involved and the time of detection. In this work, we extend the types of high-level events that can be observed at the segment-level and propose a more refined way of correlating them. Ultimately, we take a look at the underlying process instances and explore whether associations exist.

3 Preliminaries

Definition 1 (Power set, Sequence, Suffix). *Given a set A, $\mathcal{P}(A)$ is the* power set *of A and A^* are the finite sequences over A. For any $s, s' \in A^*$, we say $s' = \langle a'_1, ..., a'_m \rangle$ is a* suffix *of $s = \langle a_1, ..., a_n \rangle$ (denoted $s' \preceq s$) if and only if there is some $i \in \{0, ..., n-m\}$ such that for $j = 1, ..., m$ it holds that $a'_j = a_{i+j}$.*

Definition 2 (Events, Event log). *$\mathcal{U}_{ev}$ is the* universe of events *and Act, $Case$, Res are the sets of* activity names, case identifiers *and* resource names, *respectively. T is the totally ordered set of* timestamps. *$L = (E, Attr, \pi)$ is an* event log *where $E \subseteq \mathcal{U}_{ev}$ is a finite set of events, $\{act, case, res, time\} \subseteq Attr$ is a set of attribute names and $\pi \in E \times Attr \nrightarrow Val$ a (partial) function that assigns each event e a value $\pi(e, att)$ or is undefined (written $\pi(e, att) = \perp$). For any $e \in E$, $\pi(e, act) \in Act$, $\pi(e, case) \in Case$, $\pi(e, res) \in Res$, and $\pi(e, time) \in T$.*

For any attribute $att \in Attr$, we write $att(e)$ instead of $\pi(e, att)$ when the event log is clear from the context. Moreover, we assume that any two events of the same case never have identical timestamps.

Definition 3 (Traces, Steps, Segments). *The cases of an event log $L = (E, Attr, \pi)$ are $cases(L) = \{case(e) \mid e \in E\}$. For any case $c \in cases(L)$ with corresponding event set $E_c = \{e \in E \mid case(e) = c\}$, the* trace *of c is the sequence $\sigma(c) = \langle e_1, ..., e_{|E_c|} \rangle \in E_c^*$ containing all events from E_c ordered by time, i.e., $\forall_{1 \leq i < j \leq |E_c|}\ time(e_i) < time(e_j)$. A* step *is a pair of directly following events in a case in L. More precisely, the steps of L are $steps(L) = \{(e, e') \in E \times E \mid \exists_{c \in cases(L)}\ \sigma(c) = \langle ..., e, e', ... \rangle\}$. Moreover, we define $S(L) = \{(act(e), act(e')) \mid (e, e') \in steps(L)\}$ as the* segments *of L.*

A step is a pair of events which happened directly after each other in the same case. A segment is a pair of activities that directly follow each other in the log.

Definition 4 (Framing, Time Windows). *A* framing *is a function $\phi \in T \to \mathbb{N}$ mapping timestamps to numbers such that $\forall_{t_1, t_2 \in T}\ t_1 < t_2 \Rightarrow \phi(t_1) \leq \phi(t_2)$. Each $w \in rng(\phi)$ represents time window $\overrightarrow{w} = [w_{start}, w_{end}]$, where $w_{start} = min\{t \in T \mid \phi(t) = w\}$ and $w_{end} = max\{t \in T \mid \phi(t) = w\}$.*

Given an event log $L = (E, Attr, \pi)$ and a framing ϕ, set $W_{L,\phi} = \{w \in \mathbb{N} \mid min\{\phi(time(e)) \mid e \in E\} \leq w \leq max\{\phi(time(e)) \mid e \in E\}\}$ contains all time windows of L w.r.t. framing ϕ. Note that for any $e \in E$, $\phi(time(e)) = w$ whenever e occurred within $\overrightarrow{w}$.

4 Method

4.1 Detecting High-Level Behavior Using High-Level Events

The example in Sec. 1.2 illustrated three important components which comprise high-level behavior: the *type* of behavior observed, the *location* in the process where it emerges and the *time* aspect related to it. We call each pair of location and time information a *coordinate*.

Definition 5 (Coordinates). *Given log* $L = (E, attr, \pi)$ *and window set* $W_{L,\phi}$ *w.r.t framing* ϕ*, let* $W^2_{L,\phi} = \{(w_1, w_2) \in W_{L,\phi} \mid w_1 \leq w_2\}$*. The set* $CO(L, \phi) = S(L) \times (W_{L,\phi} \cup W^2_{L,\phi})$ *contains the* coordinates *of log* L *w.r.t.* ϕ*. Each coordinate* $co = (s, \theta) \in CO(L, \phi)$ *refers to a position in space (segment* s*) and time (window if* $\theta \in W_{L,\phi}$ *and window pair if* $\theta \in W^2_{L,\phi}$*).*

Each outlier observation we considered in Sec. 1.2 emerged from a specific set of steps. Which steps were involved in the observation depended on the type of behavior we were looking for. For instance, the steps in the first cloud in Fig. 1 represent the incoming load at segment (a, b) within a particular time window. Next, we conceptualize the colored clouds and the steps that are involved in them using *high-level features*. Each high-level feature consists of its type and a pattern. One can think of the type being the color of the cloud and the pattern being the function which determines which subset of steps occurring in a given coordinate may give rise to that type of feature.

Definition 6 (Pattern, Feature type). *Given a log* $L = (E, attr, \pi)$ *and framing* ϕ*, a* pattern *is a (partial) function* $\rho_{L,\phi} \in CO(L, \phi) \nrightarrow \mathcal{P}(E \times E) \times \mathbb{R}$ *which assigns a set of event pairs and a number to each coordinate of* L *and* ϕ*. A* high-level feature *w.r.t.* L *and* ϕ *is a pair* $hlf = (type, \rho_{L,\phi})$ *where* $type \in \mathcal{U}_{type}$ *is a feature type from the universe* $\mathcal{U}_{type}$ *of feature types and* $\rho_{L,\phi}$ *is its pattern.*

In the remainder, given log L and framing ϕ, we write $\rho^{type}_{L,\phi}$ to refer to the pattern of the high-level feature of type *type*.

In this work, we consider feature types *enter*, *exit*, *workload*, *handover*, *batch*, and *delay*. For each of these feature types, we now show how their patterns are determined. Let $co = (s, w) \in S(L) \times W_{L,\phi}$ be a coordinate from log L with time windows from framing ϕ. Let $I_s = \{(e, e') \in steps(L) \mid (act(e), act(e')) = s\}$ be the event pairs (steps) that traverse segment s in the process and let $I_w = \{e \in E \mid time(e) \in \overrightarrow{w}\}$ be the events that occur within time window w. Feature type *enter* is concerned with the steps that enter segment s during $\overrightarrow{w}$ and thus $\rho^{enter}_{L,\phi}(co) = (I^{enter}, val)$ where $I^{enter} = \{(e, e') \in I_s \mid e \in I_w\}$ and $val = |I^{enter}|$. Feature type *exit* is concerned with the steps that exit segment s during $\overrightarrow{w}$ and thus $\rho^{exit}_{L,\phi}(co) = (I^{exit}, val)$ where $I^{exit} = \{(e, e') \in I_s \mid e' \in I_w\}$ and $val = |I^{exit}|$. Feature type *workload* is concerned with the steps that exit segment s during $\overrightarrow{w}$ for which it is the same resource executing both activities of s. Thus, $\rho^{workload}_{L,\phi}(co) = (I^{wld}, val)$ where $I^{wld} = \{(e, e') \in I_s \mid e' \in I_w \wedge res(e) = res(e')\}$ and $val = |I^{wld}|$. Conversely, feature type *handover* is concerned with the steps that exit segment s during $\overrightarrow{w}$ for which there were two different resources executing the activities of s (and so real work handover took place). Hence, $\rho^{handover}_{L,\phi}(co) = (I^{hdo}, val)$ where $I^{hdo} = \{(e, e') \in I_s \mid e' \in I_w \wedge res(e) \neq res(e')\}$ and $val = |I^{hdo}|$.

The time aspect of steps which are handled in batches refers to two time windows. Let $(w, w') \in W^2_{L,\phi}$ be a pair of time windows and let $co = (s, (w, w')) \in S(L) \times W^2_{L,\phi}$ be a coordinate. Let $I_{w,w'} = \{(e, e') \in steps(L) \mid time(e) \in \overrightarrow{w} \wedge time(e') \in \overrightarrow{w'}\}$ be the set of event pairs (steps) where the first event occurs during w and the second event occurs during w'. Feature type *batch* is

concerned with the steps that enter segment s during $\overrightarrow{w}$ and exit s during $\overrightarrow{w'}$. Thus, $\rho^{batch}_{L,\phi}(co) = (I_s \cap I_{w,w'}, |I_s \cap I_{w,w'}|)$. Given a window distance $\delta \in \mathbb{N}$, the *delay* w.r.t. to δ is concerned with the steps that together experience a similar delay which is at least δ. I.e., $\rho^{delay}_{L,\phi}(co) = (I^{delay}, val)$ where $val = |I^{delay}|$ and $I^{delay} = I_s \cap I_{w,w'}$ if $w' - w \geq \delta$ and $\emptyset$ otherwise.

Definition 7 (High-level event). *Let L be an event log, ϕ a framing and $Type \subseteq \mathcal{U}_{type}$ a set of feature types. Let $thr \in Type \times S(L) \to \mathbb{R}$ be a function assigning a threshold to any type-segment pair. We observe* high-level event *$h = (type, co) \in Type \times CO(L, \phi)$ with $co = (s, \theta)$ if and only if $\rho^{type}_{L,\phi}(co) = (I^{type}, val)$ and $val \geq thr(type, s)$. Moreover, we call $C(h) \subseteq cases(L)$ the* cases *of h and for any $c \in cases(L)$, it holds that $c \in C(h)$ if and only if there exists a step $(e, e') \in I^{type}$ with $case(e) = c$. The set $\mathcal{H}_{L,\phi,Type,thr}$ contains all high-level events observed w.r.t. L, ϕ, $Type$, and thresholds from thr.*

For any of the six feature types described above, a high-level event of type *type* is observed at coordinate *co* if and only if the number of event pairs that comprise the pattern related to *type* at that coordinate is higher than a given threshold. Note that we propose the threshold to be determined based on both the feature type and the segment. For instance, for any segment s, one observes a high-level event of type *exit* at coordinate (s, w) whenever the number of steps leaving s during w is above the pth percentile of all numbers of steps which leave s in any given window throughout the process.

In the remainder of this work, we fix L, ϕ, *Type* and *thr* and set $\mathcal{H} = \mathcal{H}_{L,\phi,Type,thr}$.

4.2 Connecting High-Level Events

The various high-level events observed throughout the process are not independent of each other. As each high-level event relates to a time and place of occurrence, one can reason about time and space proximity. Moreover, many of the steps in the patterns that give rise to them may belong to the same cases. It seems natural to connect the three high-level events of types *enter*, *batch* and *handover* observed in Fig. 1 into a single *episode* of high-level behavior. That is because going from one high-level event to the next, one can notice that the cases involved have high overlap (1), and the first high-level event "stops" at the same place (2) and at the same time period (3) where the second one "begins". We use the terms *case overlap*, *location overlap*, and *time overlap* to refer to these connection criteria and we introduce them formally in this section.

Definition 8 (Start Spread, End Spread). *Let $h = (type, co) \in \mathcal{H}$ be a high-level event and let $\rho^{type}_{L,\phi}(co) = (I^{type}, val)$. We refer to time period*

$$start(h) = [min\{time(e) \mid (e, e') \in I^{type}\}, max\{time(e) \mid \{(e, e') \in I^{type}\}]$$

as the start spread *of h. Similarly, we refer to*

$$end(h) = [min\{time(e') \mid (e, e') \in I^{type}\}, max\{time(e') \mid \{(e, e') \in I^{type}\}]$$

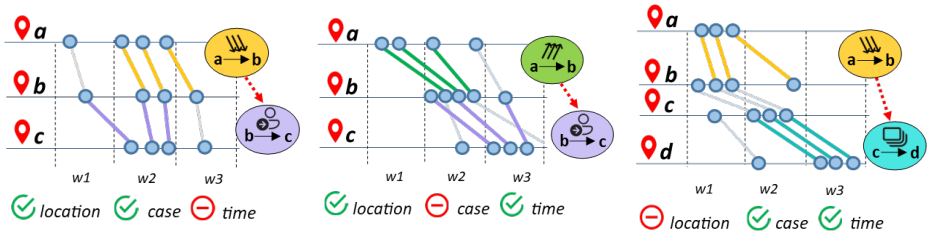


Fig. 2. For $\lambda = 0.5$, each figure shows an example where one overlap criterion from Def. 9 is not satisfied. In the left figure, high-level events $(enter, ((a,b), w_2))$ and $(handover, ((b,c), w_2)$ have no time overlap. In the middle figure, $(exit, ((a,b), w_2))$ and $(handover, ((b,c), w_3)$ have no sufficient case overlap. In the right figure, $(enter, ((a,b), w_1))$ and $(batch, ((c,d), (w_2, w_3)))$ have no location overlap.

as the end spread *of* h.

In other words, given a high-level event related to segment (a, b), the start spread covers the time period between the first and the last executions of a from the steps in the corresponding pattern. Similarly, the end spread covers the time period between the first and the last executions of b from those same steps.

Definition 9 (Overlap, Propagation). *Let* $h = (type, co), h' = (type', co') \in \mathcal{H}$ *be two high-level events with* $co = (s, \theta)$ *and* $co' = (s', \theta')$. *Given some* $\lambda \in [0, 1]$, *we say pair* (h, h') *has* case overlap *w.r.t.* λ *(denoted* $h \overset{case}{\rightsquigarrow_\lambda} h'$*) if and only if* $\frac{|C(h_1) \cap C(h_2)|}{|C(h_1) \cup C(h_2)|} \geq \lambda$. *We say pair* (h, h') *has* location overlap *(denoted* $h \overset{loc}{\rightsquigarrow} h'$*) if and only if* $s = (a, b)$, $s' = (a', b')$ *and* $b = a'$. *Moreover, we say pair* (h, h') *has* time overlap *(denoted* $h \overset{time}{\rightsquigarrow} h'$*) if and only if either* $end(h_1) \subseteq start(h_2)$ *or* $start(h_2) \subseteq end(h_1)$. *Ultimately, we say there is* propagation *from* h *to* h' *w.r.t.* λ *(denoted* $h \rightsquigarrow_\lambda h'$*) if and only if the pair* (h, h') *has case overlap w.r.t.* λ, *location overlap and time overlap. More precisely:*

$$\forall_{h,h' \in \mathcal{H}}\ h \rightsquigarrow_\lambda h' \ \Leftrightarrow\ h \overset{case}{\rightsquigarrow_\lambda} h' \ \wedge\ h \overset{loc}{\rightsquigarrow} h' \ \wedge\ h \overset{time}{\rightsquigarrow} h'.$$

Two high-level events have time overlap whenever the end spread of the first one is contained in the start spread of the second one or the other way around. Fig. 2 depicts examples of high-level event pairs that satisfy two of the overlap criteria, but not the third one. Whenever a pair of high-level events are close in time and space and their cases overlap sufficiently, we assume that their observations are correlated and we say that the first high-level event propagates to the second one. Subsequent pairs of high-level events for which propagation occurs can lead to sequences which we call *high-level episodes*.

Definition 10 (High-level episode). *Given high-level event set* $\mathcal{H}$ *and some case overlap threshold* λ, *any sequence of high-level events* $\varepsilon = \langle h_1, ..., h_n \rangle \in \mathcal{H}^*$ *creates a* high-level episode *if and only if* $\forall_{1 \leq i < n}\ h_i \rightsquigarrow_\lambda h_{i+1}$ *and* $\bigcap_{h \in \varepsilon} C(h) \geq \lambda$. *The set* $\mathcal{E}(\mathcal{H}, \lambda)$ *contains all such high-level episodes.*

In order to reason about recurring behavior, we abstract from the time component describing the high-level event and focus instead only on the type and location of the corresponding observation (the *high-level activity*). Moreover, we lift this concept to episodes and call the projection of an episode onto its high-level activities a *high-level path*.

Definition 11 (High-level path). *Let $\mathcal{H}$ be a set of high-level events and $\lambda \in [0,1]$. For any $h = (type, co) \in \mathcal{H}$ with $co = (s, \theta)$, the* high-level activity *of h is $h\uparrow = (type, s)$. For any episode $\varepsilon = \langle h_1, ..., h_n \rangle \in \mathcal{E}(\mathcal{H}, \lambda)$, the sequence $\varepsilon\uparrow = \langle h_1\uparrow, ..., h_n\uparrow \rangle$ is its corresponding* high-level path. *Multiset $P(\mathcal{H}, \lambda) = [\varepsilon\uparrow \mid \varepsilon \in \mathcal{E}(\mathcal{H}, \lambda)]$ contains all such high-level paths.*

Note that while each episode is unique because the high-level events are unique, the high-level paths may be recurring for different episodes. It is for these recurring paths that we want to investigate the correlation with the properties of the cases involved.

4.3 Case Participation in High-Level Behavior

The participating cases of a given high-level path are those which are involved in all high-level events of an episode that executes the corresponding path. The non-participating cases are those which are not participating, but which traverse the process segments underlying the path throughout their process run.

Definition 12 ((Non-)participating cases). *Given high-level event set $\mathcal{H}$ and case overlap threshold λ, let $p = \langle h_1, ..., h_n \rangle \in P(\mathcal{H}, \lambda)$ be a high-level path and for each $i \in \{1, .., n-1\}$, let $s_i = (a_i, a_{i+1})$ be the segment where the high-level event h_i was observed. The* participating cases *of p are $C_p = \{c \in cases(L) \mid \exists_{\varepsilon \in \mathcal{E}(\mathcal{H},\lambda)}\ \varepsilon\uparrow = p\ \wedge\ c \in \bigcap_{h \in \varepsilon} C(h)\}$. The* non-participating cases *of p are $\overline{C_p} = \{c \in cases(L) \setminus C_p \mid \sigma(c) = \langle e_1, ..., e_k \rangle\ \wedge\ \langle a_1, a_2, ..., a_n \rangle \preceq \langle act(e_1), ..., act(e_k) \rangle\}$.*

To measure the correlation of a case-level attribute and a high-level path, the set of cases $C_p \cup \overline{C_p}$ is additionally partitioned according to the chosen case attribute value (see Table 4.3). The correlation is then computed using the χ^2 test of independence on these two partitions. The χ^2 test measures the difference between the observed and expected frequencies for each combination of the values of two categorical variables. The null hypothesis states that there is no relationship between case participation in a given high-level path and the chosen case attribute. We consider the correlation as being statistically significant, and thus reject the null hypothesis, if the corresponding p-value of the result is smaller than 0.05.

5 Evaluation

To evaluate our method, we used the BPI Challenge 2017 log[3], which corresponds to a loan application process performed in a financial institution. Each case in

[3] https://data.4tu.nl/articles/dataset/BPI_Challenge_2017/12696884

Table 1. Given some high-level path p, the participating and non-participating case sets C_p and $\overline{C_p}$ are further split based on the chosen categorical attribute values (here: category 1 and category 2). The correlation between the attribute and the high-level path is computed using the χ^2 test of independence on the row partition (the chosen case-level attribute) and on the column partition ((non-)participation in the high-level path).

Case-level attribute	**Participating** C_p	**Non-participating** $\overline{C_p}$
category 1	n_1	n_2
category 2	n_3	n_4
	$n_1 + n_3 = \lvert C_p \rvert$	$n_2 + n_4 = \lvert \overline{C_p} \rvert$

this event log is an application. The applications can result in being successful or unsuccessful. Moreover, the duration of applications varies from less than 10 days to over 30 days. In this section, we analyze which sequences of high-level activities are strongly associated with the case attributes *outcome* and *throughput time*. In the following, we briefly describe the event log, the setup of our experiments and some general statistics over the detected high-level events. Afterwards, we comment on some of the high-level paths which showed a statistically significant correlation with the outcome and the throughput time of cases. The method together with the evaluation script is available as a Python implementation[4].

5.1 The BPI Challenge 2017 Event Log

The Application log of the BPIC 2017 contains a total of 31509 applications from January 2016 to February 2017. The general control-flow of an application can be described as follows: first, a request for a loan is made. Then, the submitted application is assessed. If a credit offer can be made, the bank composes the offer and sends it to the customer. Some time later, a bank employee calls the customer to remind them about the offer and answer any possible questions. The customer sends all necessary documents to the bank which in turn has to validate them. If the documents are incomplete, the customer is informed by a bank employee that that they have to resend the documents. If the documents are complete, the bank either composes another offer, or it makes the ultimate decision to either accept or deny the application. The activities of this process are divided into three categories: *Application State Changes* (preceded by A_), *Offer State Changes* (preceded by O_), and *Workflow Events* (preceded by W_). The workflow events additionally contain *lifecycle information* (e.g. schedule, start, suspend, resume, complete, ate_abort/withdraw). As the steps where cases move from one state of a workflow event to another make up for a significant amount of waiting time and rework in the process, we classify workflow related activities using both the event type and its lifecycle information. This results in activities like W_Call after offers|SUSPEND or W_Call after offers|RESUME.

[4] https://github.com/biankabakullari/hlem-framework

Table 2. The total number of observed high-level events in the loan application log. For each feature type, one can see the absolute and relative number of high-level events of that type, the number of distinct segments where those high-level events were observed and the segment where they were observed most often.

feature type	**# hle (%)**	**# distinct segments**	**most frequent segment**
workload	1103 (20,82 %)	36	(W_Call incomplete files|schedule, W_Call incomplete files|start)
handover	214 (4,04%)	10	(W_Call incomplete files|suspend, W_Call incomplete files|resume)
enter	1394 (26,31%)	43	(A_Create Application, A_Submitted)
exit	1377 (25,99%)	43	(A_Create Application, A_Submitted)
batch	1048 (19,78%)	11	(W_Call after offers|suspend, W_Call after offers|ate_abort)
delay	162 (3,06%)	5	(W_Validate application|suspend, W_Validate application|resume)

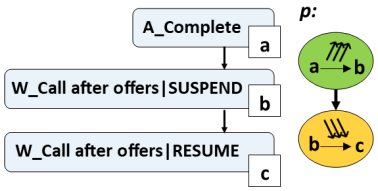
5.2 Experimental Setting and General Statistics

Before applying our method on the event log, we projected the traces onto the 43 most frequent segments. We split the time scope of the event data onto 398 windows, each corresponding to exactly one day. We evaluated the patterns related to feature types *enter*, *exit*, *workload*, *handover*, *batch* and *delay* throughout the entire coordinate space. Moreover, for *workload* and *handover*, we only considered human resources. Here, it means that we removed User_1 which was a system resource, and considered only real employees of the bank (User_2 to User_149). Each observation counted as a high-level event whenever the measured number in the corresponding pattern was at least as high as the 90th percentile of all the values obtained for that same type-segment pair. For delays, we chose a δ based on the 70th percentile of the days it takes to traverse a particular segment. This resulted in a total of 5298 high-level events. Table 5.2 shows the absolute and relative frequencies of high-level events for each feature type, together with the number of distinct segments where that type of high-level events was observed. One can notice how the activities related to application files being incomplete (W_Call incomplete files), the communication with the customers (W_Call after offers) and the application validation (W_Validate application) are most often subject to high-level behavior.

We connected the high-level events into episodes using a case overlap threshold of $\lambda = 0.5$. This generated 102060 episodes, which corresponded to 68538 distinct high-level paths. For these paths, we investigated the correlation with the *outcome* and *throughput time* of cases.

5.3 Outcome: Success Rate

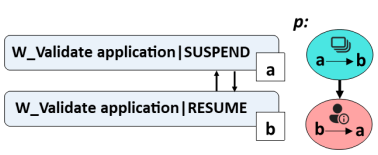
The *success rate* refers to the number of times an application results in a positive outcome (customer accepts an offer and the loan is granted) divided by the total



high exit load high enter load

outcome	C_p	$\overline{C_p}$
successful	1087 (51,06%)	11106 (53,48%)
unsuccessful	1042 (48,94%)	9661 (46,52%)
Total: 22896	$\lvert C_p\rvert = 2129$	$\lvert\overline{C_p}\rvert = 20767$

Fig. 3. The participating (C_p) and non-participating ($\overline{C_p}$) cases of the high-level path $p = \langle(exit, (a, b)), (enter, (b, c))\rangle$ where a = A_Complete, b = W_Call after offers|SUSPEND, and c = W_Call after offers|RESUME. This path was observed 14 times in the event log. Here, $\chi^2 \approx 4,55$ and $p = 0,0329$.



batching high workload

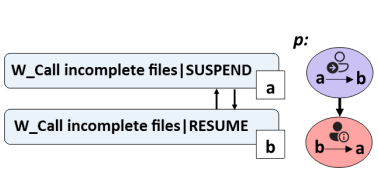
outcome	C_p	$\overline{C_p}$
successful	368 (72,73%)	7160 (77,93%)
unsuccessful	138 (27,27%)	2028 (22,07%)
Total: 9694	$\lvert C_p\rvert = 506$	$\lvert\overline{C_p}\rvert = 9188$

Fig. 4. The participating (C_p) and non-participating ($\overline{C_p}$) cases of the high-level path $p = \langle(batch, (a, b)), (workload, (b, a))\rangle$ where a = W_Validate application|SUSPEND and b = W_Validate application|RESUME. This path was observed 10 times in the event log. Here, $\chi^2 \approx 7,48$ and $p = 0,0063$.

number of applications. In our event log, a *successful* case translates into its trace containing activity A_Pending. In total, 17228 (54.85%) cases are successful and the other 12183 (45.15%) cases are unsuccessful. The latter are the cases where the loan is either denied by the bank or cancelled by the customer.

Next, we show four frequent high-level paths which showed a statistically significant correlation with the case success rate. For the two possible outcomes of *success*, a significant correlation is observed whenever $\chi^2 \geq 3.841$.

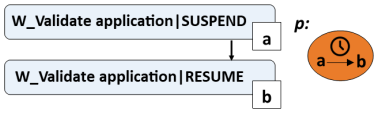
The path in Fig. 3 shows that the success rate is lower for the cases which in large groups simultaneously go from having completed the application into the part where the bank initiates communication with them. Moreover, it seems that for many cases in the process, the activities W_Validate application and W_Call incomplete files are first suspended, then resumed and afterwards suspended again. Suspending after resuming seems to be associated with high resource workload (both paths in Fig. 4 and 5). This high workload is preceded by batching behavior (Fig. 4) and high work handover ratio (Fig. 5). Participation in both these high-level paths seems to also be negatively associated with the case success rate. Additionally, cases whose validation is resumed in batches with a long waiting time after the suspension (Fig. 6) seem to also show lower success rates than the cases whose validation is suspended and then later resumed with a shorter period in-between.



high work handover ratio high workload

outcome	C_p	$\overline{C_p}$
successful	852 (79,04%)	10668 (85,05%)
unsuccessful	226 (20,96%)	1875 (14,95%)
Total: 13621	$\lvert C_p\rvert = 1078$	$\lvert\overline{C_p}\rvert = 12543$

Fig. 5. The participating (C_p) and non-participating ($\overline{C_p}$) cases of the high-level path $p = \langle(handover, (a, b)), (workload, (b, a))\rangle$ where a = W_Call incomplete files|SUSPEND and b = W_Call incomplete files|RESUME. This path was observed 16 times in the event log. Here, $\chi^2 \approx 27,54$ and $p \approx 1,53 \cdot 10^{-7}$.



delay

outcome	C_p	$\overline{C_p}$
successful	928 (72,22%)	7822 (76,81%)
unsuccessful	357 (27,78%)	2362 (23,19%)
Total: 11469	$\lvert C_p\rvert = 1285$	$\lvert\overline{C_p}\rvert = 10184$

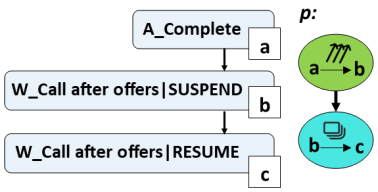
Fig. 6. The participating (C_p) and non-participating ($\overline{C_p}$) cases of the high-level path $p = \langle(delay, (a, b))\rangle$ where a = W_Validate application|SUSPEND and b = W_Validate application|RESUME. This path was observed 66 times in the event log. Here, $\chi^2 \approx 13,28$ and $p = 0,0002676$.

5.4 Throughput Time

The throughput time of a case is the time elapsed between the case's first and last event. According to [12], one can notice clear trends in the progress among the applications which take between 10-30 days to complete, and the applications which finish faster or pass the 30 days mark. We use these throughput time categories to analyze the influence of our high-level paths on case duration. In total, 7454 (23,73%) cases finish in less than 10 days, 12963 (41,27%) cases take between 10 and 30 days, and 10994 (35,00%) cases spend longer than 30 days in the process.

Next, we show three high-level paths which showed a strong association with the case throughput time. While it is unsurprising that high-level problems delay the progress of cases, our analysis reveals more detailed insights into what type of subsequent problems emerging at which process segments show particularly high association with the case duration. For the three throughput time categories, a significant correlation is observed whenever $\chi^2 \geq 5.991$.

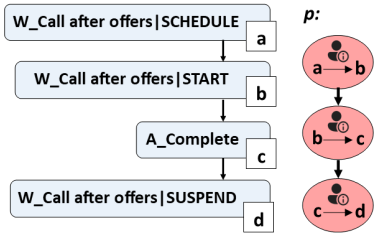
Similarly to Fig. 3, the path in Fig. 7 shows that the cases which simultaneously transition from having completed the application into a part where the bank attempts to communicate with them in batches, also suffer longer throughput times. One can notice that there are more cases that pass the 30 days' mark and less cases that finish in under 10 days from the group of participating cases than from the group of non-participating cases. Moreover, having overloaded employees taking care of the communication process with the customers (the



high exit load batching

thr. time	C_p	$\overline{C_p}$
$\leq$ *10 days*	273 (13,75%)	3860 (18,46%)
10-30 days	848 (42,72%)	8984 (42,96%)
$\geq$ *30 days*	864 (45,53%)	8069 (38,58%)
Total: 22898	$\lvert C_p \rvert = 1985$	$\lvert \overline{C_p} \rvert = 20913$

Fig. 7. The participating (C_p) and non-participating ($\overline{C_p}$) cases of the high-level path $p = \langle (exit, (a, b)), (batch, (b, c)) \rangle$ where a = A_Complete, b = W_Call after offers|SUSPEND, and c = W_Call after offers|RESUME. This path was observed 15 times in the event log. Here, $\chi^2 \approx 33,61$ and $p \approx 5,04 \cdot 10^{-8}$.



high workload

thr. time	C_p	$\overline{C_p}$
$\leq$ **10 days**	224 (18,73%)	6709 (23,20%)
10-30 days	550 (45,99%)	12006 (41,53%)
$\geq$ **30 days**	422 (35,28%)	10197 (35,27%)
Total: 30108	$\lvert C_p \rvert = 1196$	$\lvert \overline{C_p} \rvert = 28912$

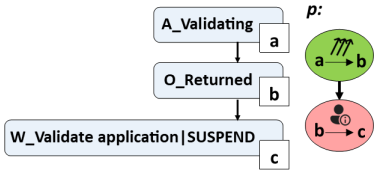
Fig. 8. The participating (C_p) and non-participating ($\overline{C_p}$) cases of the high-level path $p = \langle (workload, (a, b)), (workload, (b, c)), (workload, (c, d)) \rangle$ where a = W_Call after offers|SCHEDULE, b = W_Call after offers|START, c = A_Complete, and d = W_Call after offers|SUSPEND. This path was observed 15 times in the event log. Here, $\chi^2 \approx 15,47$ and $p = 0,000437$.

path in Fig. 8) also correlates with longer case throughput times (especially as the ratio of cases finishing in under 10 days decreases). The path in Fig. 9 covers the scenario when a case is validated, an offer is returned, but then the validating process has to be suspended. It seems that for the cases which experience batching and high workload in this process part, the throughput time worsens.

To conclude, we noticed that participation in high-level behavior was negatively associated with both *outcome* and *throughput time* as the participating cases showed lower success rates and higher throughput times compared to the non-participating cases.

6 Conclusion

In this work, we aimed to explore the interplay between high-level problems in the process and the process instances which underlie them. These problems were related to observations of high loads, busy resources, batching behavior and delays in particular locations in the process throughout different points in time. We conceptualized each single outlier observation as a high-level event and we connected these high-level events into episodes whenever they were close enough



high exit load high workload

thr. time	C_p	$\overline{C_p}$
$\leq$ **10 days**	448 (26,18%)	5067 (30,28%)
10-30 days	1019 (59,56%)	9542 (57,02%)
$\geq$ **30 days**	244 (14,26%)	2125 (12,70%)
Total: 18445	$\lvert C_p \rvert = 1711$	$\lvert \overline{C_p} \rvert = 16734$

Fig. 9. The participating (C_p) and non-participating ($\overline{C_p}$) cases of the high-level path $p = \langle (exit, (a, b)), (workload, (b, c)) \rangle$ where $a =$ A_Validating, $b =$ O_Returned, and $c =$ W_Validate application|SUSPEND. This path was observed 14 times in the event log. Here, $\chi^2 \approx 13,40$ and $p \approx 0,000123$.

in time, space, and when the cases giving rise to them were similar. For a given sequence of (outlier) observations, we wanted to investigate whether the cases that participate in that high-level behavior differ significantly from the cases which do not. For the comparison to be as meaningful as possible, the control group contained only the cases which were similar to the participating cases from a control-flow perspective. Our experiments showed that for the loan application process, there were several examples of high-level behavior at particular segments which were negatively associated with the case outcome (loan application success rate) and throughput time.

While the same method can be applied w.r.t. any process property at the case-level, the discovered significance in the connection between the emerging process behavior and the process instances underlying it is bidirectional. In future work, one could explore the cause-effect relationship behind these correlations. For *intrinsic* case properties (e.g., credit score of applicant), one could argue that it is the property itself which triggers certain kinds of high-level behavior. For *extrinsic* case properties (such as throughput time, or assigned resource for a specific task), a cause-effect discussion needs to consider the time when that property's value was set and when the high-level behavior emerged. Moreover, we cannot exclude the presence of confounding variables, that is, process aspects which influence both case participation and the case characteristic considered.

A further improvement to the method could be in the automatic segment selection. The experiments showed that some particular activities are run subsequently in an automatic way, so that reasoning about e.g., delays or work handover for these activity pairs makes less sense. Moreover, one could extend the method with a way of evaluating and ordering the detected high-level behavior by how surprising or interesting it is for the process at hand. Lastly, this method could be integrated in an interactive tool where the user selects the case property as well as high-level feature types and as a result, a list of most significant and interesting high-level behaviors w.r.t. that property is shown.

References

1. van der Aalst, W.M.P.: Process Mining: Data science in Action. Tech. rep. (2014)
2. Bakullari, B., van der Aalst, W.M.P.: High-level event mining: A framework. In: ICPM (2022)
3. Bozorgi, Z.D., Teinemaa, I., Dumas, M., La Rosa, M., Polyvyanyy, A.: Process mining meets causal machine learning: Discovering causal rules from event logs. In: ICPM (2020)
4. Bozorgi, Z.D., Teinemaa, I., Dumas, M., Rosa, M.L., Polyvyanyy, A.: Prescriptive process monitoring for cost-aware cycle time reduction. In: ICPM (2021)
5. Denisov, V., Belkina, E., Fahland, D., van der Aalst, W.M.P.: The performance spectrum miner: Visual analytics for fine-grained performance analysis of processes. In: BPM (2018)
6. Denisov, V., Fahland, D., van der Aalst, W.M.P.: Unbiased, fine-grained description of processes performance from event data. In: BPM (2018)
7. Dubinsky, Y., Soffer, P., Hadar, I.: Detecting cross-case associations in an event log: toward a pattern-based detection. Software and Systems Modeling (2023)
8. Klijn, E.L., Fahland, D.: Performance mining for batch processing using the performance spectrum. In: BPM Workshops (2019)
9. Klijn, E.L., Fahland, D.: Identifying and reducing errors in remaining time prediction due to inter-case dynamics. In: ICPM (2020)
10. Martin, N., Pufahl, L., Mannhardt, F.: Detection of batch activities from event logs. Information Systems **95**, 77–92 (2021)
11. Pika, A., Ouyang, C., ter Hofstede, A.: Configurable batch-processing discovery from event logs. ACM Transactions on Management Information Systems **13**, Article number: 28 (2022)
12. Rodrigues, A.M.B., Almeida, C.F.P., Saraiva, D.D.G., Felipe, B., Moreira, Spyrides, G.M., Varela, G., Krieger, G., Igor, T., Peres, Dantas, L.F., Lana, M., Alves, O.E., França, R., Ricardo, Neira, A., Gonzalez, S.F., Fernandes, W., Barbosa, S.D.J., Poggi, M., Lopes, H.C.V.: Stairway to value : mining a loan application process (2017)
13. Senderovich, A., Beck, J., Gal, A., Weidlich, M.: Congestion graphs for automated time predictions. Proceedings of the AAAI Conference on Artificial Intelligence **33**, 4854–4861 (2019)
14. Senderovich, A., Francescomarino, C.D., Maggi, F.M.: From knowledge-driven to data-driven inter-case feature encoding in predictive process monitoring. Information Systems **84**, 255–264 (2019)
15. Senderovich, A., Weidlich, M., Gal, A., Mandelbaum, A.: Queue mining for delay prediction in multi-class service processes. Information Systems **53**, 278–295 (2015)
16. Suriadi, S., Wynn, M., Xu, J., van der Aalst, W., ter Hofstede, A.: Discovering work prioritisation patterns from event logs. Decision Support Systems **100**, 77–92 (2017)
17. Toosinezhad, Z., Fahland, D., Köroglu, Ö., van der Aalst, W.M.P.: Detecting system-level behavior leading to dynamic bottlenecks. In: ICPM (2020)
18. van Hulzen, G.A., Li, C.Y., Martin, N., van Zelst, S.J., Depaire, B.: Mining context-aware resource profiles in the presence of multitasking. Artificial Intelligence in Medicine **134**, 102434 (2022)
19. Wimbauer, A., Richter, F., Seidl, T.: Perrcas: Process error cascade mining in trace streams. In: Process Mining Workshops (2022)