Unbiased, Fine-Grained Description of Processes Performance from Event Data

Vadim Denisov[1], Dirk Fahland[1], and Wil M.P. van der Aalst[1,2]

[1] Eindhoven University of Technology, The Netherlands,
[2] Department of Computer Science, RWTH Aachen, Germany,
`v.denisov@tue.nl`, `d.fahland@tue.nl`, `wvdaalst@pads.rwth-aachen.de`

Abstract. Performance is central to processes management and event data provides the most objective source for analyzing and improving performance. Current process mining techniques give only limited insights into performance by aggregating all event data for each process step. In this paper, we investigate process performance of all process behaviors without prior aggregation. We propose the *performance spectrum* as a simple model that maps all observed flows between two process steps together regarding their performance over time. Visualizing the performance spectrum of event logs reveals a large variety of very distinct *patterns of process performance* and performance variability that have not been described before. We provide a taxonomy for these patterns and a comprehensive overview of elementary and composite performance patterns observed on several real-life event logs from business processes and logistics. We report on a case study where performance patterns were central to identify systemic, but not globally visible process problems.

Keywords: process mining, performance analysis, visual analytics

1 Introduction

Performance analysis is an important element in process management relying on precise knowledge about actual process behavior and performance to enable improvements [11]. Descriptive performance analysis has been intensively studied within process mining, typically by annotating discovered or hand-made models with time-related information from event logs [3, 2, 1, 23] as illustrated in Fig. 1(left). These descriptive models provide aggregate measures for performance over the entire data such as average or maximum waiting times between two process steps. Models for predicting waiting times until the next step or remaining case duration learned from event data distinguish different performance classes or distribution functions based on case properties [4, 5, 12, 15].

However, these techniques assume the timed-related observations to be taken from stationary processes that are executed in isolation, i.e., that distribution functions describing performance of a case do not change over time and do not depend on other cases. These assumptions are often made by a lack of a more precise understanding of the (changes in) process performance across cases and over time.

In this paper, we consider the problem of descriptive analytics of process behavior and performance *over time*. In particular, we aim to provide a *comprehensive* description

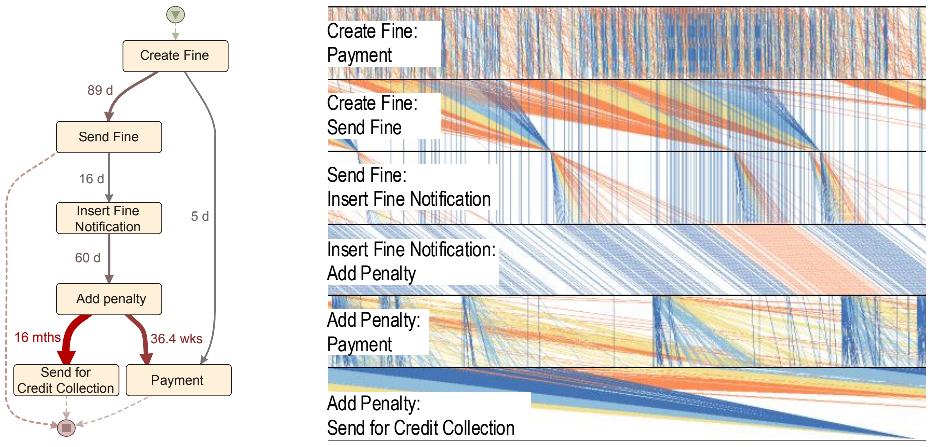

Fig. 1. Performance analysis using a graph-based model (left) and the performance spectrum (right); see online version for colored figures.

of raw process behavior without enforcing prior aggregation of data. Note that Fig. 1(left) only shows aggregates performance and no temporal patterns or changes over time. The objective of this comprehensive description is to identify patterns, trends, and properties of interest without the representational bias of an algorithm or a particular formal model.

We approach the problem through *visual analytics* which employs structuring of data in a particular form that, when visualized, allows offloading the actual data processing to the human visual system [7] to identify patterns of interest for subsequent analysis. We propose a new simple model for event data, called the *performance spectrum* and a corresponding visualization. Figure 1(right) shows the performance spectrum of the data used to the discover the model in Fig. 1(left) over a 20month period. The performance spectrum describes the event data in terms of *segments*, i.e., pairs of related process steps; the performance of each segment is measured and plotted for any occurrences of this segment over time and can be classified, e.g., regarding the overall population.

The visualization in Fig. 1(right) shows that different cases perform very differently due to systematic and unsystematic *variability of performance in the different steps over time* and synchronization of multiple cases. We implemented this visualization in an *interactive* exploration tool (ProM package “Performance Spectrum”) allowing for zooming, filtering, and performance classification and aggregation of the data.

Exploring the performance spectrum of real-life logs typically reveals numerous, novel patterns in process performance and behavior as shown in Fig. 1(right) that cannot be seen in process models as in Fig. 1(left). To enable documenting and conceptualizing these patterns for further analysis, we propose a *taxonomy* for describing elementary patterns in the performance spectrum. We evaluated the performance spectrum and the taxonomy on 12 real-life logs of business and logistics processes. Numerous elementary patterns as well as larger patterns composed of elementary ones recur throughout different event logs. We show how these patterns reveal novel insights into the interplay of control-flow, resource, and time perspective of processes. The performance spectrum of real-life logs reveals that performance in a case may be dependent on the performance of other cases, performance generally varies over time (non-stationary), and many processes exhibit temporary or permanent concept drift. We report on a case study performed with Vanderlande Industries to identify control-flow problems in very large logistics processes. Further, we found that each process has a characteristic *signature* of the patterns in its performance spectrum and that similar signatures indicate extremely similar processes not only in control-flow but also in the performance perspective.

The remainder of this paper is structured as follows. We discuss work related to performance analysis in Sect. 2. We formally define the performance spectrum in Section 3 and introduce the taxonomy for patterns in the performance spectrum in Sect. 4. We report on our evaluation on real-life event logs in Sect. 5 and discuss our findings and future work in Sect. 6.

2 Related Work

Analysis of process performance from *event data* can be divided into descriptive, predictive, and visual analysis, which we summarize here; see [12] for an extensive discussion.

Commonly, process performance is described by *enhancing* a given or discovered process model with information about durations of activities (nodes in a model) or waiting times between activities (edges in a model) [1]. In the visualization, each node and edge can be annotated with one aggregate performance measure (avg., sum, min, max) for all cases passing through this node or edge, as illustrated in Fig. 1(left). Visualization of performance on a model is more accurate if the discovery algorithm takes the underlying performance information into account [8, 18]. A non-fitting log can be aligned to a model to visualize performance [2]. More detailed visualization of performance requires more dimensions. Wynn et al. [23] plot different process variants (with different performance) into a 3-dimensional space "above" the process model. Transition system discovery allows to split occurrences of an activity based on its context and visualize performance each context separately [3].

Performance prediction for the *remaining time* until completion of a given case can be predicted by regression models [5], by annotating transition system states with remaining times [4], by learning a clustering of transition system states [6], by combining models for prediction of the next activity in a case with regression models [12]. Completion time of the next activity can be predicted by training an LSTM neural network [22], or by learning process models with arbitrary probability density functions for time delays through non-parametric regression from event logs [14] that can also be used for learning simulation models to predict performance [17, 15]. These models predict performance of a single case based on case-specific features. Performance of cases synchronizing on shared resources can be analyzed through simulation models [16] or from queuing models [19] learned from event logs. Synchronization in batch activities can be studied through queue models [13], or through aggregating event logs into a matrix [10].

The above techniques assume that probability densities for time delays are *stationary* for the whole process (do not change over time) or only depend on the individual case (*isolation* between cases). Techniques for *describing the performance of all cases* construct simpler models through stronger aggregation [3]. Also, the recent temporal network representation abstracts non-stationary changes in performance over time [18]. Techniques for predicting *performance of a single case* construct more complex models for higher precision, e.g., [22]; precision increases further when more assumption are dropped, e.g., different distribution functions [15], queues [19], non-stationarity [12]. No current model learning technique can describe process performance without making assumptions about the data. However, the results of this paper show that in particular stationarity and isolation of cases do not hold in the performance perspective.

Assumptions and representational bias of models can be avoided through visualization and visual analytics [7]. Dotted Chart [21] plots all events per case (y-axis) over time (x-axis) allowing to observe arrival rates and seasonal patterns over time. Story graphs [20] plot a case as poly-line in a plane of event types (y-axis) and time (x-axis) allowing to observe patterns of similar cases wrt. behavior and performance over time but convolutes quickly with many crossing lines.

In Sect. 3 we propose a model and visualization that avoids the problems of [20] in describing the *performance of each process step* without assumptions about the data (except having a log of discrete events). The visualization shall *reveal where a process violates typical assumption about performance* such as non-stationarity or cases influencing each other; we provide a *taxonomy* to describe these phenomena in Sect. 4.

3 Performance Spectrum

We first establish some basic notations for events and logs, and then introduce our model to describe the performance of any observable dynamic process over time.

Let A be a set of *event classifiers*; A is usually the set of activity names, but it may also be the set of resource names, or a set of locations. Let $(T, \leq, +, \cdot, 0)$ be a totally ordered set (with addition $+$, multiplication $\cdot$, and 0) of timestamps, e.g., the rational numbers $\mathbb{Q}$. An *event* $e = (a, t) \in (A \times T)$ describes that observation a occurred at time t. A *trace* $\sigma \in (A \times T)^*$ is a finite sequence of related events. An event log $L \in \mathbb{B}((A \times T)^*)$ is a multi-set of traces. For $\sigma = \langle e_1, \ldots, e_n \rangle$, we write $|\sigma| = n$ and $\sigma_i = e_i, i = 1, \ldots, n$.

The goal of the performance spectrum is to visualize the performance of process steps over time. We call $(a, b) \in A \times A$ a *process segment* describing a step from activity a to activity b, hand-over of work from resource a to b or the movement of goods from location a to b. We first formalize the *performance spectrum* for a single process segment, and then lift this to *views* on a process.

Each occurrence of a segment (a, b) in a trace $\langle \ldots, (a, t_a), (b, t_b), \ldots \rangle$ allows to measure the time between occurrences of a and b. A histogram $H = H(a, b, L) \in \mathbb{B}(T)$ describes how often all the *time differences* $t_b - t_a$ between a and b have been observed in L. In contrast, the *performance spectrum* $\mathbb{S}(a, b, L)$ collects the actual *time intervals* (t_a, t_b) observed in L. To aid recognition of patterns, we allow users to classify each interval (t_a, t_b) wrt. other observations. The specific classification depends on the analysis at hand, for example, the actual duration $t = t_b - t_a$, or whether $t = t_b - t_a$ is in the 25%-quartile of the histogram H, or other properties such as remaining time until case completion. Generally, a *performance classification* function $\mathbb{C} \in T \times T \times \mathbb{B}((A \times T)^*) \to C$ maps any interval (t_a, t_b) into a class $\mathbb{C}(t_a, t_b, L) = c \in C$. Fig. 1 classifies intervals based on the quartile of the duration in the histogram.

Definition 1 (Detailed performance spectrum). *The* performance spectrum *of a segment* (a, b) *is the bag of all its observation intervals in a trace* σ *(in a log* L*):* $\mathbb{S}(a, b, \sigma) =$

$[(t_a, t_b) \mid \exists_{1 \le i < |\sigma|} (a, t_a) = \sigma_i, (b, t_b) = \sigma_{i+1}] \in \mathbb{B}(T \times T)$*; we lift* $\mathbb{S}$ *to* L *by bag union* $\mathbb{S}(a, b, L) = \sum_{\sigma \in L} (L(\sigma) \cdot \mathbb{S}(a, b, \sigma))$*. The* detailed *performance spectrum of a segment* (a, b) *in log* L *wrt. performance classification* $\mathbb{C}$ *is* $\mathbb{S}^{\mathbb{C}}(a, b, L) = [(t_a, t_b, c) \mid (t_a, t_b) \in \mathbb{S}(a, b, L), c = \mathbb{C}(t_a, t_b, L)] \in \mathbb{B}(T \times T \times C)$*.*

Figure 1(right) visualizes the detailed performance spectrum $S = \mathbb{S}(a, b, L)$ of six different segments. For each segment (a, b) we fix coordinates y_a and y_b on the y-axis and plot each classified observation $(t_a, t_b, c) \in \mathbb{S}^{\mathbb{C}}(a_i, b_i, L)$ as a line from (t_a, y_a) to (t_b, y_b). In Fig. 1 each line is colored based on the quartile of the duration $t_b - t_a$.

The detailed performance spectrum visualizes variability of durations in a segment across cases and time. To capture and visualize also the amount of cases of particular performance over time, we define an *aggregate performance spectrum*. We *group* segments into bins of a user-chosen *period* p depending on whether they *start*, *stop*, or are *pending* in a bin, and then aggregating the observations (t_a, t_b, c) in each bin wrt. their class c (for *finitely many* classes C), akin to relational algebra or SQL operations.

Definition 2 (Aggregated performance spectrum). *Let* $S = \mathbb{S}(a, b, L)$ *be a detailed performance spectrum with finite performance classes* $C = \{c^1, \dots, c^k\}$*. Let period* $p \in T$ *and grouping* $g \in \{start, stop, pending\}$*. The* binning *of* S *wrt.* p *and* g *is the sequence of multisets* $\langle b_0, b_1, \dots \rangle$ *such that for* $i = 0, 1, \dots$ *holds*

- $b_i = [(t_a, t_b, c) \in S \mid i \cdot p \le t_a < (i + 1) \cdot p]$ *if* $g = start$*,*
- $b_i = [(t_a, t_b, c) \in S \mid i \cdot p \le t_b < (i + 1) \cdot p]$ *if* $g = stop$*, and*
- $b_i = [(t_a, t_b, c) \in S \mid i \cdot p \le t_b \text{ and } t_a < (i + 1) \cdot p]$ *if* $g = pending$ *(the segment started before the end of the bin, and ends after the start of the bin).*

The aggregation *of* S *wrt.* p *and* g *is the sequence* $agg_g(S, p)$ *of vectors* $\langle v_0, v_1, \dots \rangle$ *where each* $v_i = (v_i^1, \dots, v_i^j, \dots, v_i^k) \in \mathbb{N}^k$ *counts how often performance class* c^j *occurred in bin* b_i*:* $v_i^j = |\{(t_a, t_b, c^j) \mid (t_a, t_b, c^j) \in b_i\}|$*. The aggregated performance spectrum of a segment* (a, b) *in a log* L *is then* $\mathbb{S}^{\mathbb{C}}_{g,p}(a, b, L) = agg_g(\mathbb{S}^{\mathbb{C}}(a, b, L), p)$*.*

An aggregated performance spectrum A of one segment (a, b) can be visualized as a series of stacked bar-charts as shown in Fig. 7 where the k-th bar starts at x-coordinate $k \cdot p$ and has width p; the bottom-line of the series of bar-charts is at y-coordinate y_b and the height of all bars is normalized wrt. $y_b - y_a$.

Visualizing the performance spectrum of multiple process segments on a 2D plane requires some compromises. As the x-axis of the plane is used for visualizing time, we can only visualize control-flow by mapping segments along the single dimension of the y-axis. This forces to visualize even alternative segments (a, b) and (b, c) in a sequential manner. To give the user control over this sequentialization we let a user specify the *(sub-)trace variants Var* that shall be mapped (one after the other) onto the y-axis as in Fig. 1(right). The notion of a *view* provides all parameters for a performance spectrum.

Definition 3 (View). *A* view $V = (Var, \mathbb{C}, g, p)$ *is a set* Var *of (sub-)trace* variants $Var = \{\sigma^0, \dots, \sigma^k\} \subseteq A^*$*, a performance classification* $\mathbb{C}$*, a grouping* $g \in \{start, stop, pending, none\}$ *and period* $p \in T$*. The* segment sequence *of variant* $\sigma^i = \langle a_1^i, a_2^i, \dots, a_{n_i}^i \rangle \in Var$ *is* $seg(\sigma^i) = \langle (a_1^i, a_2^i), \dots, (a_{n_i-1}^i, a_{n_i}^i) \rangle$*. The* segment sequence *of all variants Var is their concatenation, i.e.,* $seg(Var) = seg(\sigma^1) seg(\sigma^2) \dots seg(\sigma^k)$*.*

For example, for traces $\langle a,b,c,d,e\rangle, \langle a,b,f,d,e\rangle, \langle a,b,c,b,f,e\rangle$, the variants $Var = \{\langle b,c,d,e\rangle, \langle f,d,e\rangle\}$ yield the segment sequence $seg(Var) = \langle (b,c),(c,d),(d,e),(f,d),(d,e)\rangle$.

Let L be a log, $V = (Var, \mathbb{C}, g, p)$ be a view. The *performance spectrum* of L wrt. V with $g = none$ is the sequence of the detailed performance spectra along the segment sequence $seg(Var)$: $\mathbb{S}(L,V) = \langle \mathbb{S}^{\mathbb{C}}(a,b,L)\rangle_{(a,b)\in seg(Var)}$. The *aggregated performance spectrum* of L wrt. V with $g \neq none$ is the sequence of aggregated spectra $\mathbb{S}_{g,p}(L,V) = \langle \mathbb{S}^{\mathbb{C}}_{g,p}(a,b,L)\rangle_{(a,b)\in seg(Var)}$.

For the visualization in Figure 1, the segments $seg(Var)$ are mapped to the y-axis in order of $seg(Var)$ in equidistant steps for some length $ydist$: in the i-th segment $(a_i,b_i) = seg(Var)_i$, a_i and b_i get y-coordinates $y_{a,i} = i \cdot ydist$ and $y_{b,i} = (i+1)\cdot ydist$. By default any two consecutive segments touch at $y_{b,i} = y_{a,i+1}$; an extra gap can be added whenever $b \neq a$. Figure 7 visualizes the view of an aggregate performance spectrum.

An optimal definition of Var for a given log is outside the scope of this paper, and we assume user input. Yet we identified some principles. There are two canonical trace variants Var for views on a log L. The *minimal* variant defines the most frequent variant in L, visualizing all its process segments consecutively. The *maximal* variant includes all individual observed process segments $Var_{max}(L) = \{\langle (a,b)\rangle \mid \langle \ldots,(a,t_a),(b,t_b),\ldots\rangle \in L\}$ in no specific order. Mapping segments consecutively along the y-axis allows to follow the flow of multiple cases over time as shown in Fig. 1(right). Choices in a process can be handled by defining two alternative trace variants in the view; thereby segments (a,b) occurring multiple times in Var are replicated (with the entire performance spectrum), allowing to see the flow of the variant through this segment in the performance context of other variants. Handling loops and concurrency requires event log pre-processing. Loops can be unrolled through label refinement [9] in the log. In case of concurrency, analyzing the performance of segment (a,b) with Def. 1 requires filtering from the log all activities concurrent to a and b. In case studies with Vanderlande, $Var_{max}(L)$ in combination with a hierarchical naming scheme of events allowed to visually group and analyze related segments even in very complex processes (of 10000s of segments).

4 Performance Patterns

Performance spectra of processes may contain an overwhelming amount of information and are – for the untrained eye – more difficult to read and interpret than known visualizations. However processes with similar performance characteristics show similar *patterns* in their performance spectra, and vice versa, similar patterns mean similar performance characteristics. Such patterns introduce a higher abstraction level over 'plain' performance spectra, thereby aiding in description and analysis of performance. Next, we illustrate the idea of patterns in the performance spectrum distinguishing *elementary* and *composite* patterns. We provide a *taxonomy* of elementary patterns in Sect. 4.2. We discuss composite patterns in Sect. 4.3, but posit their systematization in future work.

4.1 Elementary Patterns

Intuitively, a performance pattern is a specific configuration of the lines and bars in a performance spectrum that (1) is visually distinct within a larger part of the spectrum,

(2) describes a particular *performance scenario* (of *multiple* cases over time), and (3) repeats when this scenario repeats. An *elementary pattern* relates to a single segment and cannot be broken down further without loss of its meaning.

The elementary pattern shown in Fig. 2 occurred in segment (*Insert Fine Notification*, *Add penalty*) of the *Road Traffic Fines Management* (RF) log[3] and consists of many parallel inclined lines of the same color, corresponding to multiple observations distributed over time. Non-crossing lines show a strict FIFO order and identical inclinations show a constant waiting time for all cases. Variation in density of the lines (and in the height of the bars of the aggregated performance spectrum) shows continuous, varying workload throughout the entire log. Patterns with such characteristics are typical for highly standardized automated activities with strict time constraints. Note that existing models describe the performance of this segment as "constant" delay of 60d.

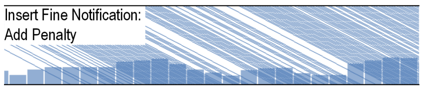

Fig. 2. The elementary pattern shows a FIFO behavior with constant waiting time

We consider the pattern to be "elementary" in the sense that we cannot decompose it further without losing its key qualities: single segment, strict FIFO with constant time, workload is continuous and varying.

4.2 Taxonomy of Elementary Patterns

We observed a great variety of elementary patterns and combinations of patterns in the performance spectra of real-life processes (see Sect. 5). That makes it impossible to provide a comprehensive catalog. Nevertheless, we are able to provide a comprehensive taxonomy of *parameters* of elementary patterns. It allows us to completely and unambiguously describe performance of a process over time in a way that patterns that correspond to similar performance scenarios have identical descriptions and identical descriptions of patterns mean similar performance scenarios, while changing the value of any parameter in a pattern would mean a different performance scenario.

The taxonomy provides parameters to characterize the Shape of lines and bars in a process in a particular Scope over time; line density and bar height describe Workload while their color describes Performance. The parameter values form a hierarchy which is shown together with typical patterns having these characteristics in Fig. 3. We provide a unique short-hand value [in brackets] for each parameter, to allow succinct notation of patterns.

Scope parameters capture the place of pattern in the performance spectrum.

– size: one segment [1 seg], one subsequence [1 sub-seq], several subsequences [>1 sub-seq]
– occurrence: globally [glob], as a local instance [loc]
– repetitions (for patterns occurring in local instance): once [once], regular [reg], periodic [per=T], arbitrary [arb],
– overlap (for repeating patterns): overlapping [overlap], non-overlapping
– duration: absolute value [D=T]

[3] `https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5`

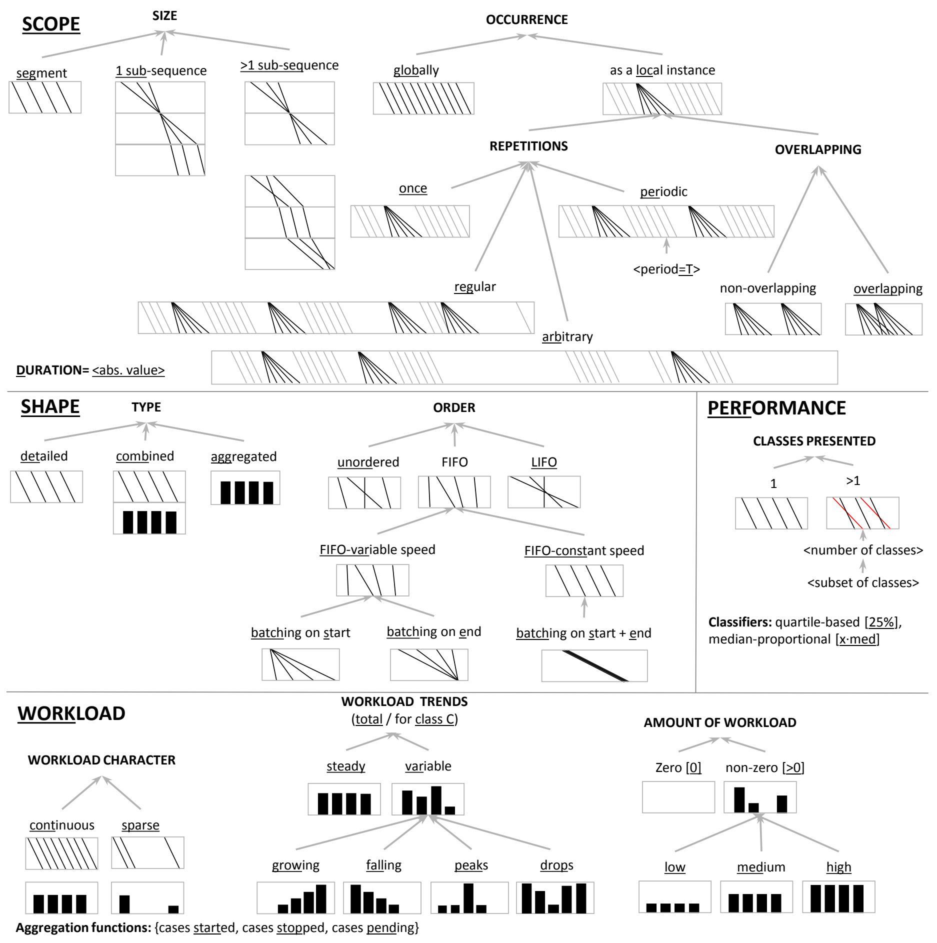

Fig. 3. Taxonomy of elementary patterns

Size describes the pattern length from the control-flow perspective: a single segment, a single subsequence or several subsequences of event classifiers. Although all elementary patterns have size 1 seg, we include this parameter in the taxonomy for compatibility with composite patterns. A pattern occurrence can be either global, when it occurs continuously throughout a segment without clear boundaries, otherwise it distinctly occurs as a local instance. Pattern instances may occur once or repeat (1) periodically in particular intervals T, (2) regularly, i.e., seemingly systematic but not periodic, or (3) arbitrarily. Repeated pattern instances can be overlapping or non-overlapping in time. Parameter duration describes the absolute duration over time (e.g. as an interval in seconds).

Shape parameters describe the appearance of lines and bars in the visualization of the performance spectrum.

– type: detailed [det], aggregated [agg], combined [comb]

– order: unordered [unord], LIFO [LIFO], FIFO with variable time [FIFO-var], FIFO with constant time [FIFO-const], batching on start [batch(s)], batching on end [batch(e)], batching on start and end [batch(s+e)]

A pattern described just in terms of lines (bars) of a detailed (aggregated) performance spectrum is detailed (aggregated); if it requires both it is combined. Order describes the configuration of lines in a detailed pattern: (1) unordered when lines irregularly cross each other, (2) LIFO when lines end in reversed order of starting, (3) FIFO when lines never cross. (3b) Non-crossing lines of variable inclination mean variable time [FIFO-var], where multiple lines starting (or ending) in a very short period show multiple cases batching on start (or on end). (3c) Lines of identical inclination show constant time [FIFO-const], where multiple lines starting and ending in a very short period (with no lines before/after) show batching on start and end.

Workload describes the height of bars in aggregated or combined patterns, and the density of lines in detailed patterns over time.

– aggregation function: segment instances started [start], stopped [stop], cases pending [pend], see Def. 2
– workload character: continuous [cont], sparse [sparse]
– amount of workload: zero [0], non-zero [>0], low [low], medium [med], high [high]
– workload trends (for a performance class or in total): can be steady [steady], variable [var], growing [grows], falling [falls], showing peaks [peak] or drops [drop]

Workload is characterized by the aggregation function defined in the view of the performance spectrum (Def. 3). Workload character can be continuous or sparse (when there are longer gaps between lines or bars), and it is visible in both detailed and aggregated patterns. Amount of workload is categorized as zero or non-zero, the latter can be categorized further as low, medium or high in relation to the maximum number of observations made on a segment (within the time period p of the view, see Def. 3). The trend over time can be steady (bars have about same height) or variable, the latter splits further into steadily growing, falling workload or showing peaks (a few high bars surrounded by lower bars) or drops.

Performance is described in terms of the performance classes present in the pattern with respect to the classifier $\mathbb{C}$ of the view (Def. 3) chosen by the user.

– classes presented: 1, > 1, number of classes, subset of classes
– Classifiers: various, we discuss quartile-based [25%] (e.g., all observations belonging to the 26%-50% quartile), median-proportional [x·med] (e.g., all observations 2-3 times longer than the median duration)

In the visualization of the performance spectrum, classes are coded by colors. A monochrome pattern has 1 class presented while a multi-colored one has > 1 classes presented.

Now we show how the taxonomy describes the elementary patterns E1-E3 found in the RF log and highlighted in Fig. 4(left). Pattern E1 occurs in a single segment in local instances with a duration of 6 months, instances repeat regularly and overlap; the

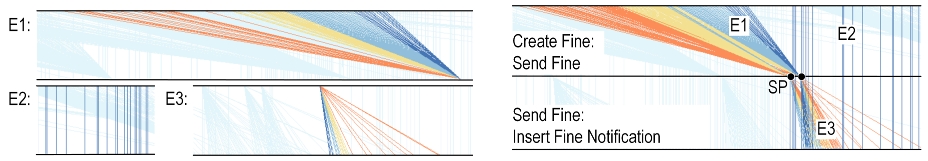

Fig. 4. Three elementary patterns E1, E2, and E3 (left) and two occurrences of a composite pattern consisting of E1-E3 (right).

detailed pattern shows batching on end in a continuous workload for 4 performance classes in a quartile-based classifier. Using the short-hand notation, we write E1 = [Scope(seg,loc,reg,overlap,D=6mo), Shape(det,batch(e)), Work(cont), Perf(25%,4 classes)]. Similarly, we can characterize E2 = [Scope(seg,glob), Shape(det,FIFO-const), Work(sparse), Perf(25%,1 class)] and E3 = [Scope(seg,loc,reg,overlap,D=1mo), Shape(det,batch(s)), Wo(cont), Perf(25%,4 classes)].

In case of creating a catalog of elementary patterns, some additional information can be added to pattern descriptions: a unique identifier and name and a meaning depending on the domain and the chosen event classifier, e.g., resources in a business process, or physical locations of a material handling system.

4.3 Composite Patterns

In the previous sections, we described performance of single process segments through elementary patterns. However, the performance spectrum of real-life processes gives rise to *composite* patterns comprised of several elementary ones. While a full taxonomy is beyond the scope of this paper, we outline some basic principles for describing composite patterns by relating elementary patterns to each other in their context.

The context of a pattern P1 as shown in Fig. 5(a) consists of (1) observations earlier and later than P1 in the same process segment, (2) observations before and after P1 in the control flow perspective, and (3) a distinct pattern P2 occurring simultaneously to P1 in the same segment. Using this context, the taxonomy can be extended with further parameters. For instance, observations before and after can be used to characterize performance of a pattern in context and the performance variants contained in the same timed period as shown in Fig. 5(b).

Figure 4(right) shows two instances of a composite pattern consisting of the elementary patterns E1, E2, and E3, described in Sec. 4.2. E1 and E3 align at a synchronization point SP, that shows synchronization of multiple cases in a "sand clock" pattern, while the cases in E2 do not synchronize with the cases in E1 or E3: we can clearly see 2 variants of behavior contained (E1+E3 and E2). The performance context of the composite pattern is diverse.

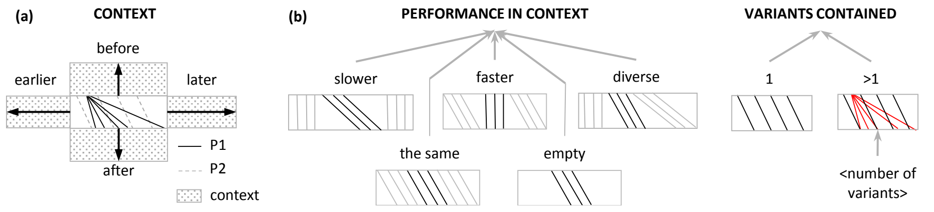

Fig. 5. (a) Pattern context, and (b) Context parameters

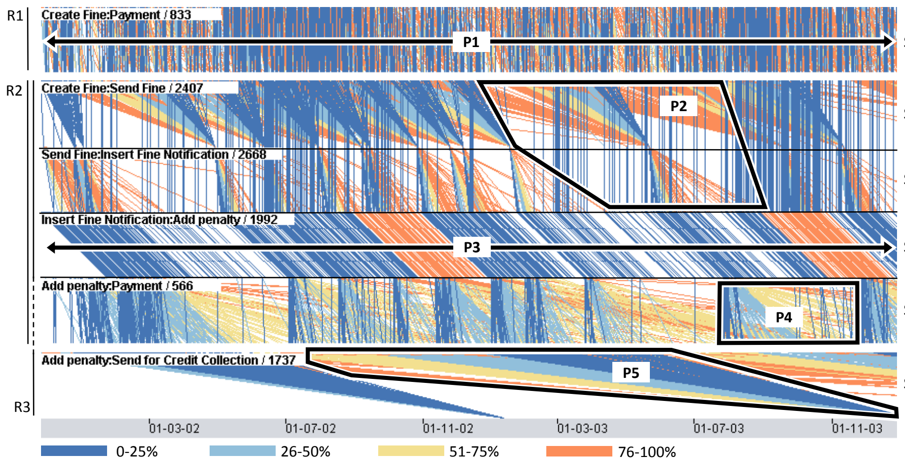

Fig. 6. Detailed performance spectrum of Road Traffic Fines Management log for years 2002 and 2003 for trace variants R1: $\langle \mathit{Create\ Fine}, \mathit{Payment} \rangle$, R2: $\langle \mathit{Create\ Fine}, \mathit{Send\ Fine}, \mathit{Insert\ Fine\ Notif.}, \mathit{Payment} \rangle$, and R3: $\langle \mathit{Create\ Fine}, \mathit{Send\ Fine}, \mathit{Insert\ Fine\ Notif.}, \mathit{Add\ penalty}, \mathit{Send\ for\ CC} \rangle$.

The taxonomy of Fig. 3 and the new parameters of Fig. 5 only partially describe composite patterns. In particular, a comprehensive taxonomy for precisely describing the alignment of patterns to each other in their context is subject of future work.

5 Evaluation

We implemented the transformation of logs into detailed and aggregated performance spectra and their visualization through an interactive ProM plug-in in package "Performance Spectrum".[4] We applied our implementation on 11 real-life event logs from business processes (BPI12, BPI14, BPI15(1-5), BPI17, Hospital Billing, RF)[5] and on 1 real-life log from logistics (BHS) provided by Vanderlande. We illustrate how the performance spectrum provides detailed insights into performance for RF; for BHS we report on a case study for identifying performance problems; and we summarize performance characteristics of the 11 business process logs.

5.1 Road Traffic Fine Management Process (RF)

Event log and view The RF event log consists of 11 activities, more than 150.000 cases and 550.000 events over a period of 12 years. We analyze the 3 trace variants R1-R3 of Fig. 6, which cover $>$ 80% of the events in the log, by defining a view for the sub-sequences $\{\langle \mathit{Create\ Fine}, \mathit{Payment} \rangle, \langle \mathit{Create\ Fine}, \mathit{Send\ Fine}, \mathit{Insert\ Fine\ Notif.}, \mathit{Add\ penalty}, \mathit{Payment} \rangle, \langle \mathit{Add\ penalty}, \mathit{Send\ for\ CC} \rangle\}$ and quartile-based performance classes.

[4] source code and further documentation available at `https://github.com/vadimmidavvv/psm`

[5] available at `https://data.4tu.nl/repository/collection:event\_logs\_real`

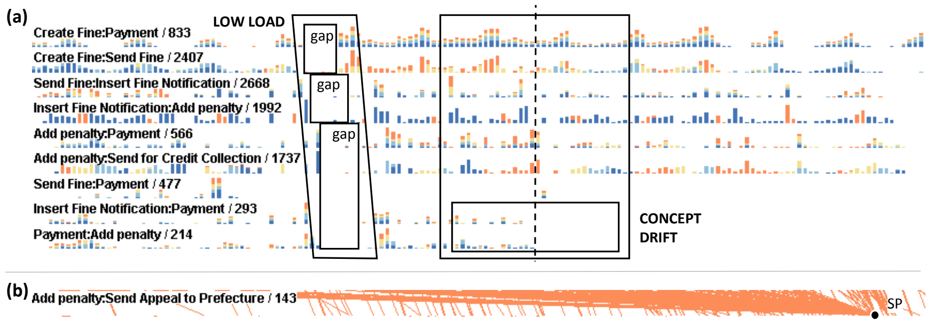

Fig. 7. Aggregated performance spectrum of Road Traffic Fines Management log (2000-2012)

First, we discuss the detailed patterns P1-P5 that can be observed in the performance spectrum of a 2-years period in Fig. 6 which represents behavior typical for the entire 12-years period. All cases start from activity *Create Fine* and continue either with activity *Payment* (variant R1) or activity *Send Fine* (R2 and R3).

P1: Segment S1 *Create Fine*:*Payment* globally contains many traces of variable duration, which are continuously distributed over time and can overtake each other, i.e., P1 = [Scope(seg,glob), Shape(det,unord), Work(cont), Perf(25%,4 classes)]. We can clearly observe that traffic offenders pay at various speeds.

P2: The performance spectrum of Fig. 6 shows that the sub-trace ⟨ *Create Fine*, *Send Fine Insert Fine notification*⟩ shared by R2 and R3 contains the composite pattern P2 which we already discussed in Sect. 4.3. P2 consists of two *different* performance variants. The "sand clock" pattern of E1+E3 of Sect. 4.1 shows that cases are accumulated over a period of 6 months; the period until *Insert fine notification* varies from zero up to 4 months. Cases in pattern E2 of Sect. 4.1 are not synchronized but processed instantly.

P3: The two variants E1+E3 and E2 vanish in the next segment S3 *Insert Fine Notif.*:*Add penalty* where all cases show a strong FIFO behavior: P3 = [Scope(seg,glob), Shape(det,FIFO-const), Work(cont), Perf(25%,2 classes)]; the switch from CEST to CET in October shows as a slower performance class in Fig. 6. After *Add penalty*, R2 continues with *Payment* (S5 in Fig. 6) and R3 continues with *send for CC* (S6 in Fig. 6).

P4: On segment S5 *Add penalty*:*Payment* we surprisingly observe emergent batching on start despite the absence of batching on end in the preceding segment S4. The "sand-clock" batching in P2 results in groups of "fast" cases which are forwarded by the FIFO pattern P3 and together create a new batching on start pattern P4 (similar to E2) in segment S5 that can take months to years to complete the *Payment*.

P5: The alternative segment S6 *Add penalty*:*Send for CC* shows batching on end every 12 months for cases that entered the batch 20 to 6 months prior: P5 = [Scope(seg,loc,per= 12mo,D=20mo), Shape(det,batch(e)), Work(cont), Perf(25%,4 classes)]. The 6-month delay revealed by P5 is mandated by Italian law. A unique pattern for this process occurs in segment *Add Penalty*:*Send Appeal to Prefecture* in Fig. 7(b) where a batch on end pattern occurs only once with a duration of 10 years.

Aggregated patterns are shown in Fig. 7(a), where every bar shows how many segments start every month. Here we can see patterns related to workload, for example, in the first

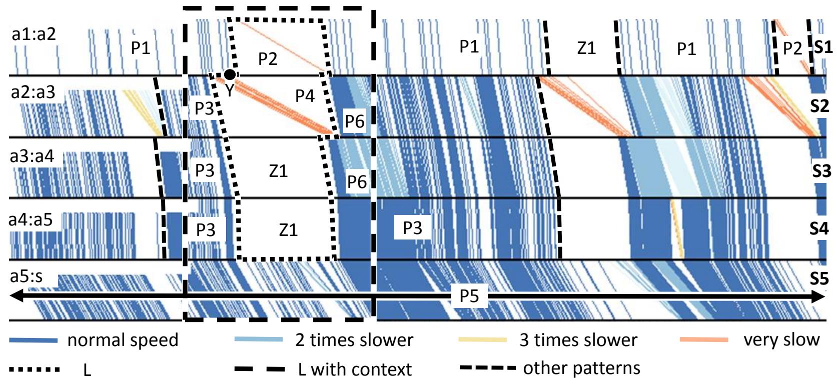

Fig. 9. Performance spectrum of bags movements between Check-In counters and the main sorter

quarter of 2004 we can see a gap pattern of 3months, gap=[Scope(seg,loc,once,D=3mo), Shape(agg,batch(e)), Work(0)]. This gap pattern propagates to subsequent segments creating a composite pattern surrounded by context with much higher load. Fig. 7(a) also reveals concept drift in the control-flow perspective: the medium non-zero workload in segments *Insert Fine Notification*:*Payment* and *Payment*:*Add penalty* drops to 0 in 2007 (up to some outliers).

5.2 Baggage Handling System of a Major European Airport (BHS)

Event Log and View In this case study, we analyzed flows of bags through a Vanderlande-built baggage handling system (BHS). In the event log, each case corresponds to one bag, events are recorded when bags pass sensors on conveyors, and activity names describe locations of sensors in the system. For 1 day of operations, an event log contains on average 850 activities, 25.000-50.000 cases and 1-2 million events.

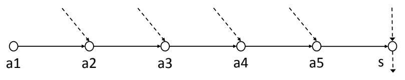

Fig. 8. The path from Check-In counter *a1* to sorter entry point *s*

To provide examples of the BHS performance spectrum and patterns, we selected conveyor subsequence $\langle a1, a2, a3, a4, a5, s\rangle$ that moves bags from Check-In counter *a1* to a main sorter entry point *s*. Cases starting in *a1* correspond to the BHS registering that a passenger put a bag onto the belt of the Check-In counter. We chose this particular part because (1) any BHS has such paths and (2) it shows many typical performance patterns of a BHS. The diagram of the corresponding system part in Fig. 8 shows that more bags join from other Check-In counters on the way in points *a2-5*. We first discuss elementary detailed patterns in the performance spectrum and then show how their compositions explain complex system behavior.

The detailed performance spectrum in Fig. 9 shows events over the period of 1 hour in a median-proportional performance classification. In the first segment S1 *a1*:*a2* we can observe pattern P1 (FIFO, constant waiting time, variable workload, normal performance) and P2 (batching on start and end with very slow performance). Empty zone Z1 shows zero workload. In BHS, FIFO behavior is typical for conveyors, where bags cannot overtake each other, and variable workload is typical for manual operations:

Table 1. Presence of selected pattern classes in real-life event logs.

	BPI12	BPI14	BPI15-1	BPI15-2	BPI15-3	BPI15-4	BPI15-5	BPI17	Hospital	H-Billing	Road Fine
unord,low		glob	glob	glob	glob	glob	gob		glob	glob	glob
unord,high								glob		glob	glob
FIFO								glob		glob	glob
FIFO+unord									reg	glob	
FIFO (weekly)	glob	glob			arb			glob			
batching		arb						per		per	reg
workload spikes								arb			reg
concept drift		once	once	once	arb	arb	arb			once	reg
sparse work	reg	reg	glob*	glob*	glob*	glob*	glob*		glob	glob	

a counter's arrival process depends on a passenger flow and their service times, which vary from passenger to passenger. Despite conveyors having constant speed S1 shows not only P1 but also P2 and Z1: some conveyors were temporarily stopped and all bags on them 'share' the same delay, as in P2.

By looking at S1 alone, we cannot explain causes of the delays in those pattern instances. But as segments in a BHS are synchronized through movement of physical objects on conveyors, we can identify the cause by following the control-flow of Fig. 8. After P4 in S2 we observe Z1 in S3 and S4, both having non-zero workload earlier (P3) and later (P3,P6), followed by non-zero workload P5 in S5 (FIFO, constant waiting time, high workload, normal performance). This gives rise to pattern L and its context highlighted in Fig. 9. Reading pattern L from S4 backwards gives the following interpretation: the conveyors in S3 and/or S4 stopped operation, so bags from S2 could not move further to S3. When S2 was stopped, S1 also was stopped (point Y), because bags could not enter S2. The slow cases of P2 and P4 are the bags waiting on a stopped conveyor. This is called a *die-back* scenario, where delays or non-operation (in S3,S4) propagate backwards through the system. When S3 and S4 return to operations, waiting bags of S1 and S2 (and from other parts that are not included in Fig. 9) resume their movement. The two times slower performance in P6 shows that S2 and S3 are at their capacity limits in this restart phase until all workload decreased. Figure 9 shows that Pattern L repeats regularly during the day.

Using the same reasoning as explained above, we identified the root cause of critical performance problems in the BHS of a large European airport which could not be identified with existing process mining tools. Our analysis took one week and was confirmed as correct by experts of Vanderlande who required several weeks of manual data analysis and years-long experience to identify the root cause.

5.3 Comparison of Event Logs

We compared the 11 real-life business process event logs regarding the types of performance patterns they contain. We visualized the performance spectrum of each log and noted the properties of the immediately visible patterns (in terms of the taxonomy of Sect. 4.2), see `https://github.com/vadimmidavvv/psm` for details.

Table 1 shows the results. We identified combined patterns of unordered behavior with low and high workload; detailed patterns of FIFO behavior, also overlaid with an unordered variant, FIFO+unord, and occurring only Mon-Sat, FIFO(weekly), and various forms of batching. The aggregate patterns showed workload spikes, concept drift, and sparse work.

The cells in Table 1 indicate for each log the occurrence and repetition values of the patterns according to the taxonomy of Fig. 3. The logs differ strongly in the presence and repetition of patterns, indicating that very different performance scenarios occur in these processes. Interestingly, the BPI15 logs which all relate to the same kind of process that is being executed in different organizations all show very similar patterns: glob* for sparse work means that sparse work co-occurs globally in a synchronized way: a large number of segments show behavior during exactly the same days.

6 Conclusion

In this paper, we proposed the performance spectrum as a novel visualization of process performance data in event logs. We project each process step (from one activity to the next) in a log over time. By making time explicit and avoiding aggregation, the performance spectrum reveals non-stationarity of performance and synchronization of different cases over time. We provided a taxonomy to isolate and describe various performance phenomena in terms of distinct elementary and composite patterns. Applying the technique on 12 real-life event logs validated its usefulness in exploration of data for identifying expected and unusual performance patterns and in confirming that process performance is neither stationary nor are cases isolated from each other. Future research is to automatically identify performance patterns from event logs and annotating process models with identified patterns. We believe the insights obtained through visual analysis to be useful in further research on performance prediction: improve queueing-based predictions based on FIFO-related patterns, aid discovery and identification of batching activities, aid in developing improved prediction models, simulation models, and prescriptive models that incorporate insights on non-stationary, or cross-case conformance checking of performance models. The identified patterns suggest also the need for performance-based filtering and sorting of event data.

Our technique is currently limited by the fact that process logic has to be flattened into sequences along the *y*-axis of the visualization, lack of support for concurrency and choices, and the very large variety of composite patterns cannot be described well by our taxonomy. Future work comprises the extension of the taxonomy, enhancement of process models with performance patterns, identifying "optimal" views for a particular analysis questions, and improved visualizations to handle concurrency and choices.

Acknowledgements. The research leading to these results has received funding from Vanderlande Industries in the project "Process Mining in Logistics". We thank Elena Belkina for support in the tool development.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery 2(2), 182–192 (2012)

3. van der Aalst, W.M.P., Pesic, M., Song, M.: Beyond process mining: From the past to present and future. In: CAiSE (2010)
4. van der Aalst, W.M.P., Schonenberg, H., Song, M.: Time prediction based on process mining. Inf. Syst. 36, 450–475 (2011)
5. van Dongen, B.F., Crooy, R.A., van der Aalst, W.M.P.: Cycle time prediction: When will this case finally be finished? In: OTM Conferences (2008)
6. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: OTM Conferences 2012. LNCS, vol. 7565, pp. 287–304. Springer (2012)
7. Keim, D.A., Andrienko, G.L., Fekete, J., Görg, C., Kohlhammer, J., Melançon, G.: Visual analytics: Definition, process, and challenges. In: Information Visualization, LNCS, vol. 4950, pp. 154–175. Springer (2008)
8. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Using life cycle information in process discovery. In: BPM Workshops 2015. LNBIP, vol. 256, pp. 204–217. Springer (2015)
9. Lu, X., Fahland, D., van den Biggelaar, F.J.H.M., van der Aalst, W.M.P.: Handling duplicated tasks in process discovery by refining event labels. In: BPM 2016. LNCS, vol. 9850, pp. 90–107. Springer (2016)
10. Martin, N., Swennen, M., Depaire, B., Jans, M., Caris, A., Vanhoof, K.: Retrieving batch organisation of work insights from event logs. Decision Support Systems 100, 119–128 (2017)
11. Maruster, L., van Beest, N.R.T.P.: Redesigning business processes: a methodology based on simulation and process mining techniques. Knowl. Inf. Syst. 21(3), 267–297 (2009)
12. Polato, M., Sperduti, A., Burattin, A., de Leoni, M.: Time and activity sequence prediction of business process instances. Computing pp. 1–27 (2018)
13. Pufahl, L., Bazhenova, E., Weske, M.: Evaluating the performance of a batch activity in process models. In: BPM Workshops 2014. LNBIP, vol. 202, pp. 277–290. Springer (2014)
14. Rogge-Solti, A., van der Aalst, W.M., Weske, M.: Discovering stochastic Petri nets with arbitrary delay distributions from event logs. In: BPM Workshops 2013. LNBIP, vol. 171, pp. 15–27. Springer (2014)
15. Rogge-Solti, A., Weske, M.: Prediction of business process durations using non-markovian stochastic petri nets. Inf. Syst. 54, 1–14 (2015)
16. Rozinat, A., Mans, R.S., Song, M., van der Aalst, W.M.P.: Discovering simulation models. Inf. Syst. 34, 305–327 (2009)
17. Senderovich, A., Rogge-Solti, A., Gal, A., Mendling, J., Mandelbaum, A., Kadish, S., Bunnell, C.A.: Data-driven performance analysis of scheduled processes. In: BPM 2015. LNCS, vol. 9253, pp. 35–52. Springer (2015)
18. Senderovich, A., Weidlich, M., Gal, A.: Temporal network representation of event logs for improved performance modelling in business processes. In: BPM 2017. LNCS, vol. 10445, pp. 3–21. Springer (2017)
19. Senderovich, A., Weidlich, M., Gal, A., Mandelbaum, A.: Queue mining for delay prediction in multi-class service processes. Inf. Syst. 53, 278–295 (2015)
20. Shrestha, A., Miller, B., Zhu, Y., Zhao, Y.: Storygraph: Extracting patterns from spatio-temporal data. In: ACM SIGKDD Workshop IDEA'13. pp. 95–103. ACM (2013)
21. Song, M., van der Aalst, W.M.: Supporting process mining by showing events at a glance. In: Proceedings of the 17th Annual Workshop on Information Technologies and Systems (WITS). pp. 139–145 (2007)
22. Tax, N., Verenich, I., Rosa, M.L., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: CAiSE 2017. LNCS, vol. 10253, pp. 477–492. Springer (2017)
23. Wynn, M.T., Poppe, E., Xu, J., ter Hofstede, A.H.M., Brown, R., Pini, A., van der Aalst, W.M.P.: Processprofiler3d: A visualisation framework for log-based process performance comparison. Decision Support Systems 100, 93–108 (2017)