Software and Systems Modeling (2026) 25:371–412
https://doi.org/10.1007/s10270-025-01321-1

SPECIAL SECTION PAPER

Enhancing explainability in process variant analysis: a framework for detecting and interpreting control-flow changes

Ali Norouzifar[1] · Majid Rafiei[1] · Marcus Dees[2] · Wil van der Aalst[1]

Received: 30 November 2024 / Revised: 9 June 2025 / Accepted: 30 July 2025 / Published online: 9 October 2025

Abstract
Processes often exhibit significant variability, posing challenges for process discovery and insight extraction. While most studies focus on detecting variability over time (e.g., concept drift), control-flow variability can also manifest across other dimensions, such as case durations or performance metrics. Identifying and understanding these changes is vital for uncovering inefficiencies and undesired behaviors. This paper introduces a novel framework that combines control-flow change detection across performance dimensions with explainability, providing insights into where and how control flow evolves. The framework uses a sliding window approach with the earth mover's distance to detect behavioral shifts. To enhance interpretability, event logs are encoded into a feature space defined by declarative constraints, capturing intuitive control-flow properties. Clustering these features reveals distinct behavioral patterns and their evolution along performance dimensions, linking detected changes to specific process dynamics. We validate the framework using three real-life event logs, including one from the UWV employee insurance agency in the Netherlands, demonstrating its ability to uncover meaningful changes, explain process variability, and support data-driven decision-making. The framework is implemented as an open-source tool for broader applicability.

Keywords Process mining · Process comparison · Business process improvement · Explainable process analysis

1 Introduction

Process mining leverages event data recorded by information systems to analyze, monitor, and improve organizational workflows. By bridging the gap between data and processes, it enables organizations to gain insights and optimize performance. Core techniques such as event log exploration, process discovery, conformance checking, and process enhancement help uncover how processes are actually executed and where improvements are possible.

Real-world processes often exhibit substantial control-flow variability, with execution behaviors differing across groups of process instances based on their characteristics. In business processes, multiple dimensions can be analyzed, with time being the most universally available due to the chronological nature of event logs. A well-studied form of variability is concept drift, which refers to changes in process execution over time [29]. Other dimensions, such as performance indicators, are domain-specific and vary across organizations, capturing aspects like efficiency, cost, or risk. This paper focuses on a different type of variability, termed performance-based process variability, which refers to differences in control flow observed across varying levels of a performance dimension such as case duration or risk score. Identifying this type of variability is important, as it can uncover behavioral patterns linked to poor performance. While concept drift has been extensively studied in process

Communicated by Han van der Aa and Rainer Schmidt.

This research was supported by the research training group "Dataninja" (Trustworthy AI for Seamless Problem Solving: Next Generation Intelligence Joins Robust Data Analysis) funded by the German federal state of North Rhine-Westphalia.

✉ Ali Norouzifar
ali.norouzifar@pads.rwth-aachen.de

Majid Rafiei
majid.rafiei@pads.rwth-aachen.de

Marcus Dees
Marcus.Dees@uwv.nl

Wil van der Aalst
wvdaalst@pads.rwth-aachen.de

[1] RWTH University, Aachen, Germany

[2] UWV Employee Insurance Agency, Amsterdam, Netherlands

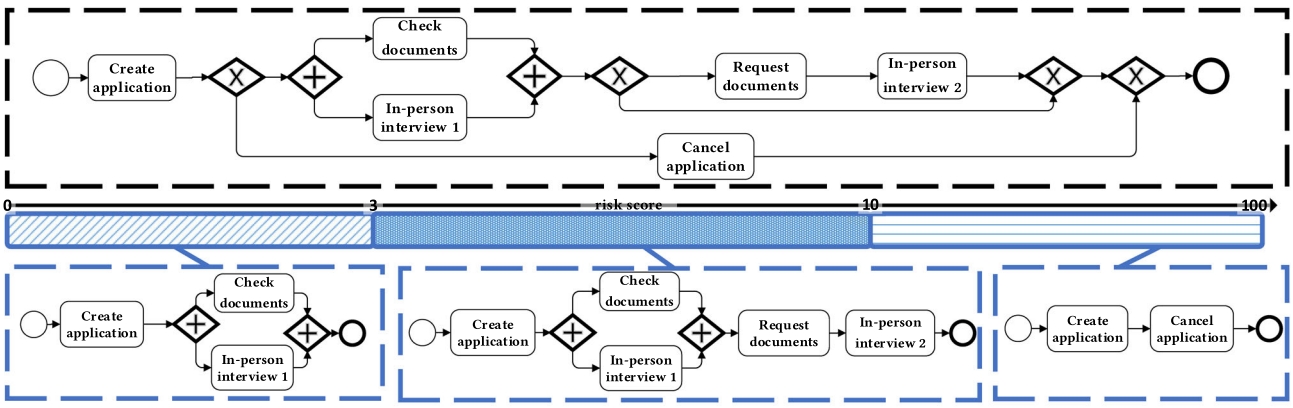


Fig. 1 BPMN representation of a claim-handling process, highlighting variations based on risk score

mining, performance-based process variability remains relatively underexplored [27].

When focusing on a particular dimension, such as risk score, our framework aims to identify key change points that segment the event log into groups with more coherent control-flow behavior. For instance, as shown in Fig. 1, the risk score assigned to each case based on the insurance company evaluations significantly influences the handling procedure. While the overarching business process model offers a high-level overview, it does not capture the variations observed in different risk score ranges. Cases with a risk score above 10 are immediately canceled after application submission. For others, the process involves steps such as document checks and interviews, with cases scoring below 3 bypassing additional document submissions and second interview. This risk score based process variability highlights the importance of a systematic approach to detecting and explaining process differences across continuous performance dimensions.

Detecting and explaining control-flow changes within such processes is essential for organizations to adapt and optimize effectively. Traditional process mining techniques focus on discovering process models, monitoring deviations, and optimizing performance [1]. However, they often overlook performance-based process variability. Performance dimensions often exist since the organizations tend to measure the performance of their processes in order to have better understanding of the deficiencies and profitability of the processes. The relation between these performance dimensions and control flow is less investigated in the literature. Identification of such relations helps to improve the way that the cases are handled and pinpoint the exact patterns that derive these differences.

This paper addresses this gap by proposing a framework that integrates explainability into performance-based process variability detection, enabling process experts to understand behavioral dynamics driving detected changes in control flow over performance dimensions. Building on the framework introduced in [27], we propose an enhanced approach. The original framework assigns an indicator value to each case and sorts the cases accordingly. A sliding window then progresses sequentially from the beginning (i.e., cases with low indicator values) to the end (i.e., cases with high indicator values). At each step, the Earth Mover's Distance (EMD) is used to compare the control-flow behavior of cases in the left half of the window with those in the right half, effectively quantifying the degree of similarity between the two groups.

Despite the effectiveness of the proposed framework in identifying *where* control-flow changes occur, the original framework lacks interpretability regarding *why* and *how* these changes arise. The absence of insights into the drivers of performance-based process variability limits its applicability for in-depth process analysis. To overcome this, our enhanced framework integrates segmentation with interpretable behavioral models, enabling process analysts to gain a deeper understanding of the dynamics underlying these changes. Compared to the previous work, our approach offers improved explainability and a more comprehensive perspective on performance-based process variability, equipping organizations with insights to better address inefficiencies and optimize workflows. The key contributions of this paper are:

1. **Control-Flow Behavioral Signal Extraction and Clustering:** A novel feature space derived from declarative constraints is introduced to represent control-flow characteristics as interpretable signals. These signals capture process dynamics across selected dimensions and are clustered into distinct behavioral patterns.
2. **Explainability-Driven Framework:** By analyzing the change points detected through the EMD-based framework in conjunction with clustered behavioral signals, we provide explainable, natural language interpretations of the specific behaviors that change. This approach

enhances the framework's interpretability and supports insights for process improvement.

3. **Extended Experiments and Comparative Analysis:** We extend our case study using real-world event data from the UWV agency to derive interpretable explanations for the detected change points. In addition, we evaluate the proposed method on several publicly available real-life event logs. To highlight the strengths and distinctive contributions of our framework, we compare the insights it generates with those obtained from adapted versions of two state-of-the-art techniques [2, 34], which primarily focus on identifying variability over time.

The remainder of this paper is structured as follows: Sect. 2 reviews the related literature. Section 3 introduces the mathematical notations used throughout the paper. Section 4 presents the components of our proposed method, detailing how control-flow changes across performance dimensions are identified and explained. A real-world case study, along with two additional experiments using publicly available event logs, is presented in Sect. 5. Finally, Sect. 6 concludes the paper and outlines directions for future research.

2 Related work

Table 1 provides an overview of the relevant studies, categorizing them based on their variant identification strategies and the nature of their explainability. We organize the related work into two main categories. The first category, *process variant identification*, includes approaches that partition event log traces into subgroups with more homogeneous control flow. These methods may be *dimension-driven*, exploring specific dimensions such as time or performance indicators, or *dimension-agnostic*, operating without predefined criteria. Additionally, we differentiate between techniques that identify process variants *automatically* and those that operate in a *semi-automatic* manner, requiring user intervention or preprocessing steps such as the discretization of continuous variables prior to variant extraction.

The second category, *explainability*, comprises methods that describe and interpret behavioral differences between trace groups. Some techniques offer *direct* explainability as part of their output, while others rely on *indirect* interpretations through visualizations or statistical summaries. As summarized in Table 1, only a few techniques simultaneously address process variant identification and provide explainability by highlighting the control-flow characteristics that best distinguish the identified variants.

2.1 Process variant identification

Real-world business processes are inherently complex due to their high variability, which often stems from case-specific characteristics [22]. To address this complexity, researchers have proposed various methods for analyzing processes from different perspectives, often conceptualized as a *multidimensional process cube*, where each dimension provides unique insights [6].

One approach to managing process variability involves statistically analyzing process variants. By deriving process variants based on specific criteria, statistical tests can be employed to determine whether significant differences exist between these variants [18]. While this method allows the analyst to choose a relevant dimension, it relies on an initial segmentation of the cases, which may not always be readily available or obvious.

Clustering techniques are extensively utilized to group similar traces and identify distinct process variants [36]. However, the selection of appropriate similarity metrics poses a significant challenge [4]. Hierarchical trace clustering methods have also been proposed, enabling the hierarchical merging of similar trace groups based on different ranges of an attribute [37]. Additionally, identifying deviant cases through trace clustering has proven effective in reducing process complexity [17]. Clustering techniques can automatically explore the event log and, without relying on a specific dimension, partition the cases into meaningful clusters.

The active trace clustering framework [33] identifies groups of traces that exhibit similar control-flow behavior by integrating process model information into the clustering process. While traditional clustering techniques typically lack built-in explainability, this approach generates a process model for each cluster, allowing analysts to manually compare the models and uncover the key behavioral differences between clusters.

Cohort analysis, leveraging the earth mover's distance, has been employed to detect process variants considering different ranges or values of an attribute [19]. While this approach is effective and supports dimension-driven analysis, it typically requires discretizing continuous attributes that may not always be straightforward or practical without prior domain knowledge.

The process variant identification method proposed in [27] uses a sliding window approach based on earth mover's distance to analyze cases sorted by a specific performance indicator, automatically identifying change points in the control flow. While this framework focusses on identifying process variants across different performance dimensions, most existing methods remain focused on analyzing changes over the time dimension, as seen in concept drift detection approaches [29]. Concept drift methods typically encode

Table 1 Overview of related work categorized by their approach to process variant identification and explainability

Research study	Process variant identification		Explainability
	Auto/semi	Dim.-driven/agnostic	Direct/indirect
Statistical tests [18]	Semi	Dimension-driven	–
Clustering [4, 17, 36, 37]	Auto	Dimension-agnostic	–
Active trace clustering [33]	Auto	Dimension-agnostic	Indirect
Cohort analysis [19]	Semi	Dimension-driven	–
EMD-based process variant identification [27]	Auto	Dimension-driven	–
Concept drift detection [8, 23, 29]	Auto	Dimension-driven	–
Explainable concept drift detection [2, 34]	Auto	Dimension-driven	Direct
Deviance mining [5, 28]	–	–	Direct
Discriminative temporal patterns [16]	–	–	Direct
Decision mining [21]	–	–	Direct
Explainable outcome prediction [9]	–	–	Direct
Comparative process mining [31]	–	–	Direct
Visualization [35]	–	–	Indirect
Explainable process variant identification (our paper)	Auto	Dimension-driven	Direct

event data into a feature space and identify change points based on feature values.

Declarative constraints, with their strong capability to encode sophisticated control-flow characteristics, have proven effective in detecting control-flow changes over time [34]. Statistical tests have also been employed to detect changes by analyzing the distributions of partially ordered runs observed in consecutive time windows [23]. Feature-based concept drift detection methods may introduce noise or bias due to feature selection. Non-feature-based approaches, such as those leveraging the earth mover's distance and sliding windows, provide an alternative by directly comparing process behaviors without relying on feature encoding [8].

2.2 Explainability

Some concept drift detection methods not only identify change points, but also offer explainability by highlighting which characteristics have changed. The visual concept drift detection approach [34] provides visualizations that assist domain experts in identifying declarative constraints whose evaluations differ across time windows where changes occur. The framework proposed in [2] investigates causal relationships between change points in individual time series, each derived from specific measured features. It employs Granger causality analysis to uncover which changes are likely to have triggered others, thus explaining the drivers behind observed drifts.

Declarative process models provide an effective alternative for representing processes with high variability, addressing the limitations of imperative notations like BPMN by capturing long-term dependencies and complex activity relationships [14]. Algorithms leveraging declarative constraints have been developed to detect process deviations, providing valuable tools for identifying and analyzing variations in behavior [5, 28].

Assuming that the event log is already segmented, the discriminative temporal patterns framework [16] provides methods for identifying declarative constraints that most effectively distinguish between segments. This framework builds on concepts from deviance mining, declarative process discovery, and explainability techniques derived from predictive modeling.

Given that a process model is available, either discovered automatically or defined by domain experts, decision mining techniques can be employed to refine decision points in the model according to different segments of cases [21]. This analysis enables explainability by revealing how specific choices at decision points influence the likelihood of a case progressing into a particular segment. From an explainability perspective, such methods enhance the interpretability of process models by explicitly linking decision points to distinct behavioral patterns or case groups.

Explainability is a key area of focus in predictive process monitoring, where significant efforts aim to interpret the decisions made by predictive models [9]. Once a model is trained to classify cases into predefined segments based on a feature set, various explainability techniques can be applied to identify which features most strongly influence the model's decisions regarding case classification. Additionally, comparing and explaining differences between process variants offers insights into the key distinctions, helping organizations identify areas for improvement [31].

Choosing appropriate visualization techniques is essential for providing domain experts with clear and explainable insights [35]. Effective visualizations enable process experts to better understand and interpret detected changes, making the analysis more intuitive. While visualization techniques may not offer direct explainability, they can still support domain experts by enabling visual identification of key differences in process behavior.

The explainable process variant identification method proposed in this paper extends the framework introduced in [27], which automatically detects change points along a performance dimension. In contrast to the original framework, our approach adds an explainability layer that clarifies what behavioral aspects have changed at each identified point. Unlike existing explainable concept drift detection methods [2, 34], which primarily focus on changes over time, our method is tailored to analyze variation along performance dimensions, offering more targeted insights for process improvement.

By identifying process variants and evaluating the desirability or undesirability of segments, the results contribute to enhancing various process analysis techniques. These include the discovery of imperative models, e.g., [25], and declarative models, e.g., [12, 30]. Declarative constraints derived from these segments can also be leveraged as valuable input for more advanced discovery algorithms [26]. Furthermore, these process variants can be used by outcome prediction methods [11, 32] and process variant comparison approaches [7]. Such insights empower organizations to identify desirable and undesirable behaviors, refine workflows, and drive data-driven decisions to optimize operational performance.

3 Preliminaries

This section provides the formal definitions and mathematical notations used throughout the paper. These definitions form the foundation for understanding the proposed framework.

A multiset extends the concept of a set by allowing multiple occurrences of the same element. $\mathcal{B}(A)$ represents the set of all multisets over a base set A. For example, $B_1 = [x^2, y]$ and $B_2 = [x, y, z^{10}]$ are two multisets over $A = \{x, y, z\}$. For $B \in \mathcal{B}(A)$, the frequency of an element $a \in A$ in B is denoted by $\#_B(a)$. We write $b \in B$ if $\#_B(b) > 0$. $|B| = \sum_{b \in B} \#_B(b)$ is the number of elements in this multiset. Considering B_1 and B_2, we can write, $\#_{B_1}(x) = 2$, $\#_{B_2}(z) = 10$, $z \notin B_1$, and $|B_2| = 12$. The power set of a set A, denoted $\mathcal{P}(A)$, is the set of all subsets of A, including the empty set and A itself. A^* is the set of all finite sequences over A. We denote the set $\{n \in \mathbb{N} \mid p \leq n \leq q\}$ by $\mathbb{N}^{[p,q]}$ for brevity.

Definition 1 *(Event Log)* Let $\mathcal{C}$, $\mathcal{A}$, and $\mathcal{T}$ denote the universe of case identifiers, activities, and timestamps, respectively. An event is a tuple $e = (c, a, t)$, where:

$$\pi_{\mathcal{C}}(e) = c \in \mathcal{C}, \quad \pi_{\mathcal{A}}(e) = a \in \mathcal{A}, \quad \pi_{\mathcal{T}}(e) = t \in \mathcal{T}.$$

The universe of all events is denoted by $\mathcal{E} = \mathcal{C} \times \mathcal{A} \times \mathcal{T}$. A trace is a finite sequence of events $\sigma = \langle e_1, e_2, \ldots, e_n \rangle \in \mathcal{E}^*$ such that:

$$\forall 1 \leq i < n : \pi_{\mathcal{C}}(e_i) = \pi_{\mathcal{C}}(e_{i+1}) \quad \text{and} \quad \pi_{\mathcal{T}}(e_i) \leq \pi_{\mathcal{T}}(e_{i+1}).$$

An event log $L \subseteq \mathcal{E}^*$ is a set of traces where each trace corresponds to a distinct case identifier. The universe of all event logs is denoted by $\mathcal{L}$.

The function $cf : \mathcal{L} \to \mathcal{B}(\mathcal{A}^*)$ extracts the control flow of an event log L, defined as the multiset of traces projected onto their activity sequences:

$$cf(L) = [\langle \pi_{\mathcal{A}}(e_1), \ldots, \pi_{\mathcal{A}}(e_n) \rangle \mid \langle e_1, e_2, \ldots, e_n \rangle \in L].$$

The activity set of an event log L is $act(L) = \{a \in s \mid s \in cf(L)\}$.

An example event log from the claim-handling process shown in Fig. 1 is detailed in Table 2. To simplify notation and avoid long activity names, we use alphabetic representations (enclosed in parentheses) that uniquely correspond to each activity name. In this event log, the mappings for event e_5 are:

$$\pi_{\mathcal{C}}(e_5) = A\_105, \quad \pi_{\mathcal{A}}(e_5) = f, \quad \text{and}$$
$$\pi_{\mathcal{T}}(e_5) = 22.09.2024.$$

According to Table 2, the corresponding event log is:

$$L_1 = \{\langle e_1, e_2, e_6 \rangle, \langle e_3, e_5 \rangle, \langle e_4, e_8, e_9, e_{11}, e_{12} \rangle, \langle e_7, e_{10} \rangle\}.$$

The control flow of this event log is represented as:

$$cf(L_1) = [\langle a, b, c \rangle, \langle a, f \rangle^2, \langle a, c, b, d, e \rangle].$$

In addition to control flow and timestamps, case-level indicators derived from other data sources can provide valuable insights into process behavior. A case-level indicator is a numerical value assigned to each case.

Definition 2 *(Case-Level Indicator)* Let $L \in \mathcal{L}$ be an event log. A case-level indicator is a function $\kappa_L : L \to \mathbb{R}$ that assigns a numerical value $\kappa_L(\sigma)$ to each trace $\sigma \in L$. If the context is clear, we omit the subscript L and write $\kappa(\sigma)$.

Table 2 Example event log extracted from the claim-handling process represented in Fig. 1

Event	Case ID	Activity	Timestamp
e_1	*A_104*	*Create application* (*a*)	10.09.2024
e_2	*A_104*	*Check documents* (*b*)	11.09.2024
e_3	*A_105*	*Create application* (*a*)	13.09.2024
e_4	*A_106*	*Create application* (*a*)	17.09.2024
e_5	*A_105*	*Cancel application* (*f*)	22.09.2024
e_6	*A_104*	*In-person Interview 1* (*c*)	23.09.2024
e_7	*A_107*	*Create application* (*a*)	24.09.2024
e_8	*A_106*	*In-person Interview 1* (*c*)	27.09.2024
e_9	*A_106*	*Check documents* (*b*)	29.09.2024
e_{10}	*A_107*	*Cancel application* (*f*)	30.09.2024
e_{11}	*A_106*	*Request documents* (*d*)	02.10.2024
e_{12}	*A_106*	*In-person Interview 2* (*e*)	22.10.2024

Table 3 Examples of case-level indicators for the claim-handling process event log represented in Table 2

Case ID	Risk score	Duration (days)	Claim amount
A_104	20	13	2500
A_105	1	9	3000
A_106	7	35	5200
A_107	63	6	800

Case-level indicators are performance metrics defined at the level of individual cases. These indicators are domain-specific and vary across organizations. For the claim-handling process event log shown in Table 2, examples of case-level indicators include *risk score*, *duration*, and *claim amount*, as detailed in Table 3. Each of these indicators evaluates specific aspects of individual cases, providing valuable insights into their characteristics and performance.

Definition 3 *(Ordering Function)* Let $L \in \mathcal{L}$ be an event log and κ be a case-level indicator. The ordering function $rank_\kappa(L)$ arranges the traces in L in non-decreasing order of their indicator values:

$$rank_\kappa(L) = \langle \sigma_1, \sigma_2, \ldots, \sigma_{|L|} \rangle,$$

where $L = \{\sigma_1, \sigma_2, \ldots, \sigma_{|L|}\}$ and $\kappa(\sigma_i) \leq \kappa(\sigma_j)$ for all $1 \leq i < j \leq |L|$.

For the example event log L_1 shown in Table 2, let κ represent the *risk score* assigned to cases. The case-level indicator κ is as follows:

$$\kappa = \left\{(\sigma_{A\_104}, 20), (\sigma_{A\_105}, 1), (\sigma_{A\_106}, 7), (\sigma_{A\_107}, 63)\right\}.$$

Using these scores, the ordering function ranks the cases as:

$$rank_\kappa(L_1) = \langle \sigma_{A\_105}, \sigma_{A\_106}, \sigma_{A\_104}, \sigma_{A\_107} \rangle.$$

To enable a comparison between event logs with varying numbers of cases, it is essential to extract their stochastic languages. This approach ensures that the measurements are independent of the event log sizes, focusing instead on the underlying behavioral patterns.

Definition 4 *(Stochastic Language)* Let $\mathcal{A}$ be the universe of activities. A stochastic language is a function $f : \mathcal{A}^* \to [0, 1]$ satisfying $\sum_{s \in \mathcal{A}^*} f(s) = 1$. The universe of stochastic languages is denoted by $\mathcal{F}$.

Definition 5 *(Stochastic Language of an Event Log)* Let $L \in \mathcal{L}$ be an event log. The *stochastic language* of L, denoted $stoch_L : cf(L) \to [0, 1]$, is the function defined by

$$stoch_L(s) = \frac{\#_{cf(L)}(s)}{|cf(L)|} \quad \text{for all } s \in cf(L).$$

For example, the stochastic language of L_1 is

$$stoch_{L_1} = \left\{ \left(\langle a, b, c \rangle, \frac{1}{4}\right), \left(\langle a, f \rangle, \frac{2}{4}\right), \left(\langle a, c, b, d, e \rangle, \frac{1}{4}\right) \right\}.$$

The earth mover's distance quantifies the dissimilarity between two stochastic languages by calculating the cost of transforming one distribution into another.

Definition 6 *(Earth Mover's Distance)* Let $L, L' \in \mathcal{L}$ be two event logs, $\delta : \mathcal{A}^* \times \mathcal{A}^* \to [0, 1]$ a trace distance function, and $r : cf(L) \times cf(L') \to [0, 1]$ a reallocation function. The Earth Mover's Distance between L and L' is:

$$EMD(L, L') = \min_{r \in \mathcal{R}} \sum_{s \in cf(L)} \sum_{s' \in cf(L')} r(s, s') \cdot \delta(s, s'),$$

subject to:

$$\sum_{s' \in cf(L')} r(s, s') = stoch_L(s),$$
$$\sum_{s \in cf(L)} r(s, s') = stoch_{L'}(s'),$$
$$r(s, s') \geq 0.$$

We use the Levenshtein distance as the trace distance function δ. The optimal reallocation r is computed using linear programming techniques [20]. A value of zero in the earth mover's distance indicates that the stochastic languages of the event logs are identical, assigning the same probability to each trace. A value of one means that the entire probability mass must be reallocated to traces that are at the maximum possible ground distance of 1 from each other.

Consider another group of cases from the claim-handling process illustrated in Fig. 1, represented by the following

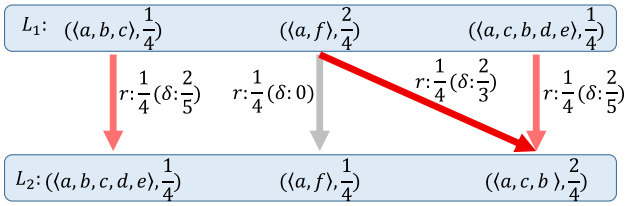


Fig. 2 Visualization of reallocation amounts and distances between traces for calculating the earth mover's distance between L_1 and L_2. The color intensity corresponds to the magnitude of the distance between the traces

stochastic language:

$$stoch_{L_2} = \left\{ \left(\langle a,b,c,d,e\rangle, \frac{1}{4}\right), \left(\langle a,f\rangle, \frac{1}{4}\right), \left(\langle a,c,b\rangle, \frac{2}{4}\right) \right\}.$$

To compute the EMD between L_1 and L_2, we first determine the pairwise distances between traces using the normalized Levenshtein distance. This metric computes the minimum number of single-element edits (insertions, deletions, or substitutions) required to transform one sequence into another, divided by the length of the longer sequence. The result is a similarity measure ranging from 0 (identical sequences) to 1 (completely different sequences of non-overlapping activities).

For example:

- The distance between $\langle a,b,c\rangle$ and $\langle a,b,c,d,e\rangle$ is $2/5 = 0.4$, since two insertions are needed and the longer sequence has length 5.
- The distance between $\langle a,f\rangle$ and itself is 0.
- The distance between $\langle a,c,b,d,e\rangle$ and $\langle a,c,b\rangle$ is $2/5 = 0.4$, due to the removal of d and e.
- The distance between $\langle a,c,b,d,e\rangle$ and $\langle a,b,c,d,e\rangle$ is $2/5 = 0.4$, requiring two substitutions.

The reallocation amounts and distances between traces are depicted in Fig. 2. Using these distances and the optimal reallocation of probability mass, the EMD is computed as the total cost of transforming one distribution into the other:

$$EMD(L_1,L_2) = \tfrac{1}{4}\times 0.4 + \tfrac{1}{4}\times 0 + \tfrac{1}{4}\times 0.666 + \tfrac{1}{4}\times 0.4 = 0.37.$$

This value reflects the minimal cost required to align the distributions over the space of traces, where cost is based on normalized structural dissimilarity.

4 Process variant identification and explanation framework

Figure 3 provides an overview of the components of our proposed method. The control-flow change detection component identifies key change points in control flow by analyzing cases ordered according to a specific performance dimension. Additionally, we integrate a feature generation and clustering framework to investigate how different control-flow characteristics change across the range of the selected performance dimension. By combining these two views, our approach offers a unique and explainable method for detecting change points, allowing not only the identification of where changes occur, but also a clear interpretation of what characteristics are changing and how.

To demonstrate the functionality of each component of our proposed method, we use the process model depicted in Fig. 1 as a running example. This model was implemented and simulated using *CPN Tools* to generate a synthetic event log, denoted as L_6. The event log L_6 comprises 10,000 cases and five trace variants[1], and serves to illustrate the step-by-step execution of our framework.

4.1 Process variant identification framework

Unlike concept drift approaches, which primarily focus on temporal dimensions, our framework identifies control-flow changes across continuous dimensions. Instead of generating a large feature set, which might miss certain types of changes, the framework leverages the earth mover's distance to effectively measure performance-based process variability.

The detection process is based on three concepts: *bucketing*, *windows*, and *local distance function*. Each concept builds upon the previous one to enable the detection of control-flow changes. We introduce these concepts formally below and provide illustrative examples to clarify their roles in the framework.

To analyze an event log, we first segment it into smaller groups, referred to as buckets. This segmentation allows the framework to compare local subsets of traces within the log.

Definition 7 [Buckets] Let $L \in \mathcal{L}$ be an event log, κ be a case-level indicator, $\sigma_p = rank_\kappa(L)(p)$ for $p \in \mathbb{N}^{[1,|L|]}$, and $b \in \mathbb{N}^{[2,|L|]}$ be the number of buckets specified by the user. For simplicity, assume that $|L|$ is divisible by b. Then, $l = \frac{|L|}{b}$ represents the number of cases in each bucket, such that:

$$B_q = \langle \sigma_{(q-1)\cdot l+1}, \ldots, \sigma_{q\cdot l}\rangle, \quad 1 \le q \le b.$$

[1] https://github.com/aliNorouzifar/X-PVI/tree/master/assets/test.xes

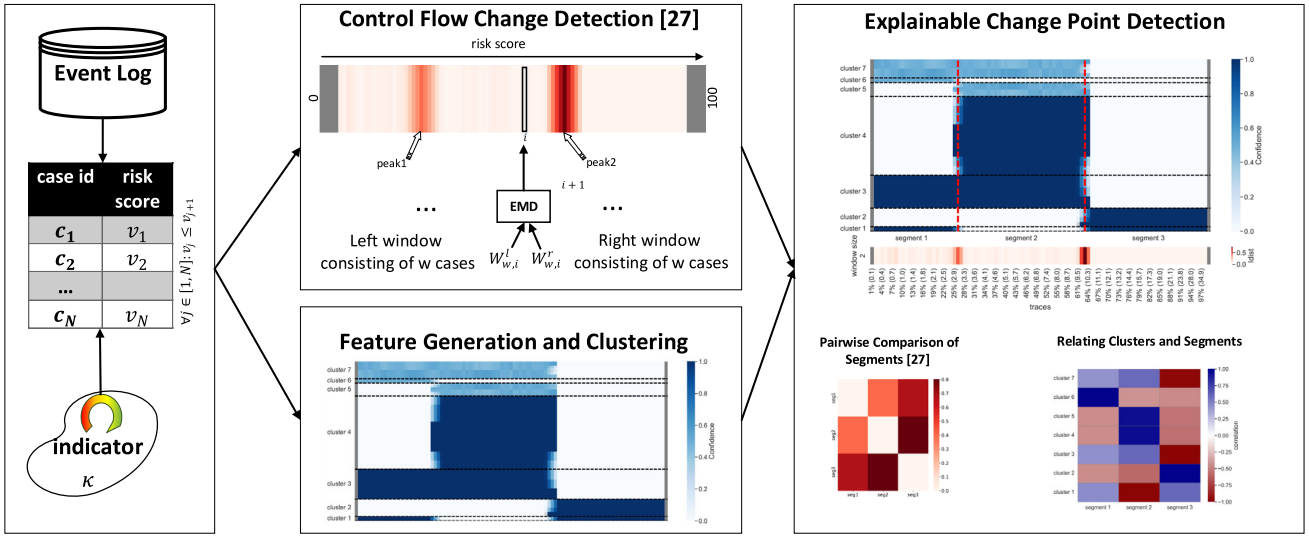


Fig. 3 Proposed framework for analyzing process variants driven by performance dimensions, such as risk scores

For example, consider the event logs L_3, L_4, and L_5, each containing 150 cases, with the following control-flow distributions:

- $cf(L_3) = [\langle a, b, d\rangle^{120}, \langle a, b, c\rangle^{30}]$
- $cf(L_4) = [\langle a, b, d\rangle^{60}, \langle a, b, c\rangle^{30}, \langle a, e, d\rangle^{60}]$
- $cf(L_5) = [\langle a, b, c\rangle^{30}, \langle a, e, d\rangle^{120}]$

Assuming a bucket size of $b = 15$, each log is partitioned into 15 buckets containing 10 traces each. Figure 4a visualizes this bucketing process for the three logs, after sorting the cases based on a performance indicator. Each event log exhibits different types of control-flow changes. In L_3, the control flow changes after the 6th bucket and then reverts to its initial distribution, resembling the pattern at the beginning. In L_4, a control-flow change also occurs after the 6th bucket, followed by another change after the 9th bucket. In L_5, the control flow shifts after the 6th bucket but returns to the original distribution shortly after the 7th bucket.

Once the event log is segmented into buckets, the next step is to create windows for comparison. A window consists of two parts: a *left window* and a *right window*, which represent buckets to the left and right of a sliding central point, respectively.

Definition 8 *(Left and Right Windows)* Let $L \in \mathcal{L}$ be an event log, κ be a case-level indicator, $\sigma_p = rank_\kappa(L)(p)$ for $p \in \mathbb{N}^{[1,|L|]}$, $b \in \mathbb{N}^{[2,|L|]}$ be the number of buckets, and $w \in \mathbb{N}^{[1,\frac{b}{2}]}$ be a window size parameter. Given a window size w and an index $i \in \{w, \ldots, b-w\}$, the left and right windows are defined as:

$$W^l_{w,i} = \{\sigma \in B_q \mid i - w < q \leq i\},$$
$$W^r_{w,i} = \{\sigma \in B_q \mid i < q \leq i + w\}.$$

For instance, with $b = 15$ and a window size of $w = 3$, the left window at position $i = 5$ would include buckets B_3, B_4, and B_5, while the right window would include buckets B_6, B_7, and B_8.

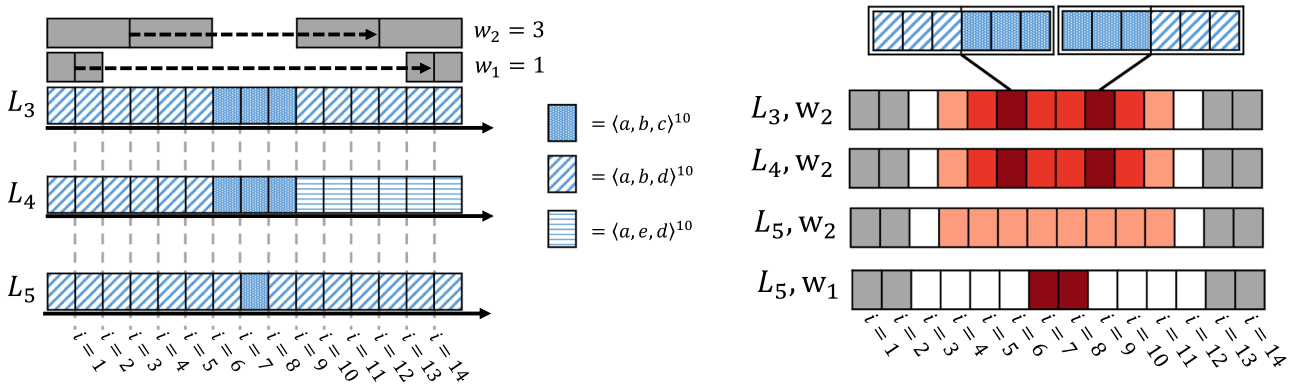
To quantify the difference between the left and right windows, we use the earth mover's distance. This function captures the degree of variability in control flow between the two windows.

Definition 9 *(Local Distance Function)* Let $L \in \mathcal{L}$ be an event log, κ be a case-level indicator, $b \in \mathbb{N}^{[2,|L|]}$ be the number of buckets, and $w \in \mathbb{N}^{[1,\frac{b}{2}]}$ be a window size parameter. The local distance function is defined as:

$$ldist_{L,\kappa,w,b} : \{w, \ldots, b-w\} \to [0, 1],$$
$$ldist_{L,\kappa,w,b}(i) = EMD(W^l_{w,i}, W^r_{w,i}),$$

where EMD represents the earth mover's distance between the left and right windows (cf. Def. 6). For brevity, $ldist_{L,\kappa,w,b}$ is simplified to $ldist$ when the context is clear.

The sliding window is moved across the range of κ, starting at $i = w$ and stopping at $i = b - w$. This ensures both left and right windows are fully defined. Figure 4b visualizes $ldist$ values for various configurations of event logs and window sizes. The intensity of the red color in the heatmap is proportional to the value of the $ldist$ function, i.e., the trace variants in the left and right windows have identical stochastic probabilities, resulting in an earth mover's distance of zero. Darker shades of red represent higher $ldist$ values, indicating greater dissimilarity between the distributions. Gray areas in the heatmap denote positions where the $ldist$ function is undefined, i.e., those points are outside the domain of the function.



(a) Visualization of event logs L_3, L_4, and L_5, each exhibiting different types of control-flow changes.

(b) Visualization of *ldist* values (cf. Def. 9) as the sliding window is applied across event logs.

Fig. 4 Examples demonstrating the framework's operation using three distinct event logs, a bucketing parameter $b = 15$, and two window sizes, $w_1 = 1$ and $w_2 = 3$

For L_3 with w_2, a behavioral change is detected at $i = 5$, with the most significant differences observed at $i = 7$ and $i = 10$. Similarly, L_4 with w_2 exhibits a comparable pattern. However, the $ldist$ metric does not indicate whether the behavior before $i = 7$ and after $i = 10$ is similar. When analyzing L_5, a smaller window size ($w_1 = 1$) detects sharp changes at $i = 7$ and $i = 10$, whereas a larger window size ($w_2 = 3$) identifies changes starting at $i = 5$ but obscures the precise change points. This demonstrates that larger windows are more robust to noise but may fail to capture critical change points. Therefore, in our analysis, we present results using different window sizes, enabling users to discern whether a detected change is superficial and noisy or represents a significant behavioral shift.

4.1.1 Running example

In Fig. 5, the cases are ordered based on their risk score. The results are shown for $b = 100$ (i.e., 1% of the cases in each bucket) and different window sizes $w \in \{2, 5, 10, 15\}$. The parenthetical numbers indicate the raw risk score values. The experiment is repeated for each window size, and the color intensity represents the magnitude of the earth mover's distance between the left and right windows, highlighting where significant changes occur.

Considering $w = 2$ in Fig. 5 and $\theta_{cp} = 0.1$, $ldist$ has two peaks in $i = 26$ ($risk\ score$=3) and $i = 63$ ($risk\ score = 10$). We can use heatmaps to compare the resulting segments pairwise to check if non-adjacent segments have similar control flows. Using the identified peaks, the whole event log is divided into three segments, and the segments are compared in Fig. 11a. The results reveal that segment 2 and segment 3 have the highest distance, followed by segment 1 and segment 3, and finally segment 1 and segment 2. The process variants depicted in Fig. 1 align with these findings, supporting the validity of the identified segments.

4.2 Behavioral signals generation and clustering

Once the segments are identified, the subsequent step is to provide process experts with insights into the types of behaviors that change across the range of the indicator and to characterize the process behavior associated with each segment. While the earth mover's distance is highly effective for detecting changes in control flow, it does not inherently explain the specific nature of these changes. To address this limitation, we introduce a novel framework that incorporates declarative constraints as interpretable representations of control-flow characteristics. By monitoring how these constraints evolve across the indicator's range, our approach adds a layer of explainability to the segmentation process.

The framework generates a comprehensive set of behavioral features based on declarative constraints and evaluates these features as the sliding window moves along the event log. Features that exhibit similar patterns of change are grouped using clustering techniques, highlighting shared behavioral dynamics.

4.2.1 Declarative constraints

Declarative constraints offer a flexible and interpretable way to model control-flow behaviors in processes. By defining behavioral rules using generic templates and instantiating them with specific activity labels, they enable an expressive representation of constraints that can be evaluated against event logs.

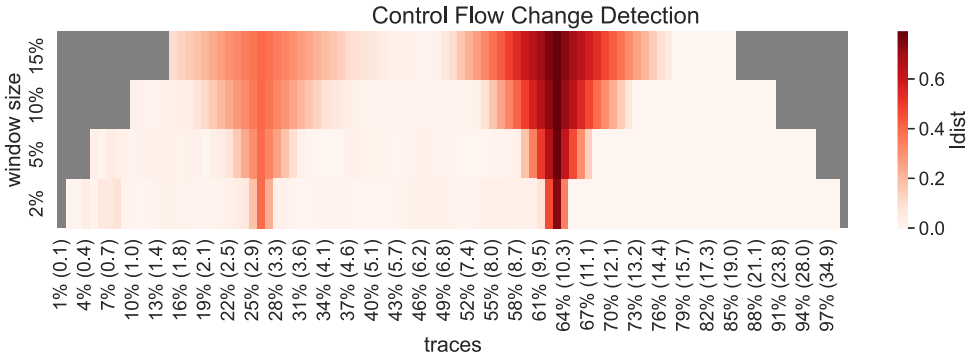


Fig. 5 Control-flow change detection using the earth mover's distance framework with b=100 and different window sizes $w \in \{2, 5, 10, 15\}$. The color intensity indicates the magnitude of control-flow changes

Definition 10 *(Declarative Templates and Constraints)* Let $\mathcal{DT}$ be a non-empty set of declarative templates, where each template is a relation $d(x_1, \ldots, x_m) \in \mathcal{DT}$ defined over variables $x_1, \ldots, x_m$, with $m \in \mathbb{N}$ representing the arity of d. For a given set of activities $a_1, \ldots, a_m \in \mathcal{A}$, a declarative constraint $d(a_1, \ldots, a_m)$ is derived as a specific instantiation of the template d, binding its variables to the corresponding activities.

A declarative constraint specifies that any trace $\sigma \in \mathcal{A}^*$ is allowed unless there is explicit evidence that the trace violates the constraint, as determined by the semantics of the associated template. Such a violation is denoted as $\sigma \not\models d(a_1, \ldots, a_m)$. The language of a declarative constraint, denoted as $Lang(d(a_1, \ldots, a_m))$, is the set of all control-flow traces that satisfy the constraint:

$$Lang(d(a_1, \ldots, a_m)) = \{\sigma \in \mathcal{A}^* \mid \sigma \models d(a_1, \ldots, a_m)\}.$$

The declarative templates used in this paper capture a range of control-flow behaviors. These templates are as follows:

- $RespondedExistence(x_1, x_2)$: If x_1 occurs, x_2 occurs as well.
- $CoExistence(x_1, x_2)$: If x_1 occurs, x_2 occurs as well and vice versa.
- $Response(x_1, x_2)$: If x_1 occurs, then x_2 occurs after x_1,
- $AlternateResponse(x_1, x_2)$: If x_1 occurs, then x_2 occurs afterward before x_1 recurs.
- $ChainResponse(x_1, x_2)$: If x_1 occurs, then x_2 occurs immediately after it.
- $Precedence(x_1, x_2)$: x_2 occurs only if preceded by x_1,
- $AlternatePrecedence(x_1, x_2)$: x_2 occurs only if preceded by x_1 with no other x_2 in between.
- $ChainPrecedence(x_1, x_2)$: x_2 occurs only if x_1 occurs immediately before it.
- $Succession(x_1, x_2)$: x_1 occurs if and only if it is followed by x_2.
- $AlternateSuccession(x_1, x_2)$: x_1 and x_2 occur if and only if they follow one another, alternating.
- $ChainSuccession(x_1, x_2)$: x_1 and x_2 occurs if and only if x_2 immediately follows x_1.
- $Absence(x_1)$: x_1 does not occur.
- $AtLeast1(x_1)$ / $AtLeast2(x_1)$ / $AtLeast3(x_1)$: x_1 occurs at least once / two times / three times.
- $AtMost1(x_1)$ / $AtMost2(x_1)$ / $AtMost3(x_1)$: x_1 occurs at most once / two times / three times.
- $End(x_1)$: x_1 is the last to occur.
- $Init(x_1)$: x_1 is the first to occur.

Definition 11 *(Declare Set of an Event Log)* Let $L \in \mathcal{L}$ be an event log. The function $declset$ extracts the set of all declarative constraints that can be instantiated over the set of activities in L. Formally,

$$declset(L) = \{d(a_1, \ldots, a_m) \mid m \in \mathbb{N} \wedge d(x_1, \ldots, x_m) \in \mathcal{DT} \wedge a_1, \ldots, a_m \in act(L)\},$$

where $declset(L)$ represents all constraints definable over the activities in the event log L.

For example, consider event log $L_3 \in \mathcal{L}$ with $cf(L_3) = [\langle a, b, c\rangle^{120}, \langle a, b, d\rangle^{30}]$, as illustrated in Fig. 4a. The set of activities in L_3 is $act(L_3) = \{a, b, c, d\}$. Examples of declarative constraints for L_3 include $\{Response(a, b), CoExistence(c, d), Init(a)\} \subset declset(L_3)$. $Response(a, b)$ is satisfied because, in all traces, every occurrence of a is followed by b, $Init(a)$ is satisfied because a is the first activity in every trace, and $CoExistence(c, d)$ is violated because c and d never appear together in the same trace.

Declarative constraints, by their rule-based nature, can be evaluated using measures such as *confidence*, *support*, and *coverage* to assess their quality [15]. These measures can be calculated at two levels: the *trace level*, where constraints are evaluated across entire process instances, and the *event level*, where individual occurrences of activities are considered. For instance, in the trace $\langle a, c, b, a, b, a\rangle$, the constraint

$Response(a, b)$ is violated because the last a is not followed by any b. However, at the event level, a occurs three times, and the constraint is satisfied for two of these occurrences while being violated once. Measuring at the event level provides a more granular perspective and prevents the overlooking of instances where the constraint is satisfied. In this paper, we adopt the event-level approach to ensure a more nuanced evaluation of constraint quality.

Each declarative constraint also has an *activation*, which specifies the conditions under which the constraint is evaluated. For example, in the constraints $Response(a, b)$ and $Precedence(a, b)$, the activations are a and b, respectively. If no activation is triggered in a trace, there is no evidence to evaluate whether the constraint is satisfied or violated; in such cases, the constraint is said to be vacuously satisfied [15]. While vacuous satisfaction can occur naturally, relying on it can lead to misleading results. To address this, we consider only instances where constraints are *non-vacuously satisfied*, ensuring that evaluations are based on meaningful activations.

Definition 12 *(Confidence and Coverage)* Let $L \in \mathcal{L}$ be an event log. For a given declarative constraint $c \in declset(L)$, let the following be defined:

- $\#_{activated}(L, c)$: The number of times the constraint c is activated in L.
- $\#_{satisfied}(L, c)$: The number of times the constraint c is satisfied in L.
- $\#_{events}(L)$: The total number of events in L.

Using these quantities, the coverage and confidence of a constraint c are defined as follows:

- $Coverage : declset(L) \to [0, 1]$ is the proportion of activated events to the total number of events in L:

$$Coverage(L, c) = \frac{\#_{activated}(L, c)}{\#_{events}(L)}.$$

- $Confidence : declset(L) \to [0, 1]$ is the proportion of satisfied events to the number of activated events in L. To avoid division by zero, the denominator is set to at least 1:

$$Confidence(L, c) = \frac{\#_{satisfied}(L, c)}{\max(1, \#_{activated}(L, c))}.$$

For example, consider the event log L_3 and the declarative constraints $c_1 = Response(a, d)$ and $c_2 = Precedence(a, d)$. The confidence and coverage metrics for these constraints are calculated as follows:

- $Coverage(L, c_1) = \frac{1\times120+1\times30}{3\times120+3\times30} = 0.33$,
  $Confidence(L, c_1) = \frac{1\times30}{1\times120+1\times30} = 0.20$.
- $Coverage(L, c_2) = \frac{1\times30}{3\times120+3\times30} = 0.07$,
  $Confidence(L, c_2) = \frac{1\times30}{1\times30} = 1.00$.

4.2.2 Feature generation

The feature generation step aims to extract meaningful features from the event log that can act as behavioral signals, highlighting changes in control-flow behavior. Each declarative constraint represents a rule or an expected behavior within the process, providing a structured way to monitor the event log. For example, a constraint may specify that certain tasks must always follow one another, capturing dependencies or patterns that indicate significant behavioral dynamics.

Definition 13 *(Behavioral Signals)* Let $L \in \mathcal{L}$ be an event log, and let κ be a case-level indicator used to order the cases, $b \in \mathbb{N}^{[2,|L|]}$ be the number of buckets given by the user, and $w \in \mathbb{N}^{[1,\frac{b}{2}]}$ be a window size parameter.

$W_{w,i} = W^l_{w,i} \cup W^r_{w,i}$ represents the union of the cases in the left and right windows at each $i \in [w, b - w]$. For a declarative constraint $c \in declset(L)$, the behavioral signal s_c is defined as:

$$s_c = \langle m(W_{w,w}, c), \ldots, m(W_{w,b-w}, c)\rangle,$$

where $m(W_{w,i}, c)$ is a measure that depends on a user-defined coverage threshold $\theta_{cvg} \in [0, 1]$ and is given by:

$$m(W_{w,i}, c) = \begin{cases} Confidence(W_{w,i}, c), & \text{if } Coverage(W_{w,i}, c) \geq \theta_{cvg}, \\ 0, & \text{otherwise.} \end{cases}$$

The set of all behavioral signals for event log L is denoted as:

$$\mathcal{S}_L = \{s_c \mid c \in declset(L)\}.$$

We leverage the Minerful framework to generate declare set of event logs and evaluate their *confidence* and *coverage* within each time window [14, 34]. Incorporating θ_{cvg} threshold on the *coverage* value is crucial to ensure the robustness of the algorithm. Without this threshold, a few activations of a constraint within a window, especially if all activations are satisfied, could result in a high confidence value. Such cases are often due to noise and may create a misleading impression of significant changes in the behavioral signals. By applying θ_{cvg}, the algorithm excludes windows with low activation counts, thereby preventing the overemphasis of spurious patterns and ensuring that the identified signals reflect meaningful process behavior.

Not all behavioral signals in $\mathcal{S}_L$ are informative for explaining changes in control-flow behavior. Some signals may have constant values across all windows, indicating no

variability and thus no meaningful contribution to the analysis. These can be safely removed. For example, a behavioral signal may always be zero if the corresponding constraint is never satisfied in any trace of the event log, or it may always be one if the constraint is never violated. Removing such constant signals helps streamline the analysis by focusing only on features that exhibit meaningful changes.

Subsumption, in the context of declarative constraints, refers to a hierarchical relationship where satisfying a more specific constraint automatically satisfies a more general one because the specific constraint imposes stricter conditions [13]. This relationship poses challenges when analyzing sets of constraints. For instance, if $ChainSuccession(a, b)$ holds for a set of traces within a window, those traces also satisfy $Succession(a, b)$, as the latter is less restrictive. When two declarative constraints yield identical confidence values, the more restrictive constraint is typically preferred due to its more precise characterization of process behavior. This principle extends to behavioral signals: if two signals are identical and represent subsumption behavior, only the signal associated with the more restrictive constraint is retained. Formally, given $c_1, c_2 \in declset(L)$, if $Lang(c_1) \subset Lang(c_2)$ and $s_{c_1} = s_{c_2}$, then c_2 is excluded from further analysis. The subsumption relationships among the declarative templates considered in our study are summarized in Fig. 6, where arrows indicate the direction from less restrictive to more restrictive constraints.

Definition 14 *(Pruning Function for Behavioral Signals)* Let $L \in \mathcal{L}$ be an event log and $\mathcal{S}_L$ be the set of all behavioral signals extracted for this event log. We define:

- $S_{constant} = \{s_c \in \mathcal{S}_L \mid \forall i \in \mathbb{N}^{[1,|s_c|)} : s_c(i) = s_c(i+1)\}$, the set of constant signals, where the signal has the same value across all windows.
- $S_{hierarchy} = \{s_c \in \mathcal{S}_L \mid \exists s_{c'} \in \mathcal{S}_L : s_c = s_{c'} \wedge Lang(c') \subset Lang(c)\}$, the set of signals corresponding to constraints that are subsumed by more restrictive constraints.

The pruning function $prune : \mathcal{P}(\mathcal{S}_L) \to \mathcal{P}(\mathcal{S}_L)$ removes non-informative signals and is defined as:

$$prune(\mathcal{S}_L) = \mathcal{S}_L \backslash (S_{constant} \cup S_{hierarchy}),$$

where the resulting set $\mathcal{S}'_L = prune(\mathcal{S}_L)$ contains only the pruned behavioral signals that are informative and non-redundant.

Figure 7 illustrates the pruning step of the feature generation framework using event log L_3 as an example. For simplicity, we consider a subset of declarative templates $\{Response(x_1, x_2), RespondedExistence(x_1, x_2), Precedence(x_1, x_2)\} \subset \mathcal{DT}$. Since $act(L_3) = \{a, b, c, d\}$, the set of possible declarative constraints and corresponding behavioral signals, $\mathcal{S}_L$, contains 36 behavioral signals. The intensity of the colors in the figure corresponds to the value of the behavioral signal, i.e., the *confidence* of the declarative constraint, provided that the *coverage* exceeds the threshold $\theta_{cvg} = 0.02$, within specific windows. Among these, 16 behavioral signals are constant zeros, as their corresponding constraints are never satisfied, e.g., $Response(d, c)$, and eight behavioral signals are constant ones, as their constraints are always satisfied, e.g., $Precedence(a, d)$. From the remaining 12 behavioral signals, 4 represent subsumption relationships and are therefore removed. For instance, between $RespondedExistence(a, c)$ and $Response(a, c)$, the more specific $Response(a, c)$ is retained. After pruning, eight behavioral signals remain: $\{RespondedExistence(d, a), RespondedExistence(d, b), Precedence(a, d), Precedence(b, d), Response(a, c), Response(a, d), Response(b, c), Response(b, d)\}$.

4.2.3 Feature clustering

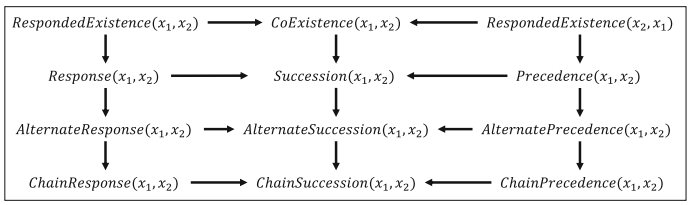
After generating the set of constraint signals and pruning them, the next step is to analyze their patterns and group similar behavioral signals. Identifying groups of features with similar change patterns provides insights into the behavioral characteristics that evolve together across the process. As illustrated in Fig. 8, some features exhibit comparable patterns of variation, suggesting shared behavioral dynamics. Clustering techniques are employed to systematically uncover these similarities and identify sets of process specifications that undergo changes together.

Definition 15 *(Behavioral Clusters)* Let $L \in \mathcal{L}$ be an event log. $\mathcal{S}'_L = prune(\mathcal{S}_L)$ represent the set of all pruned behavioral signals. A clustering algorithm is a total, surjective function $clust : \mathcal{S}'_L \to \{1, \ldots, n\}$, such that it assigns each behavioral signal $s_c \in \mathcal{S}'_L$ to one of n clusters. The i-th cluster, denoted as $clust_i$, satisfies the following properties:

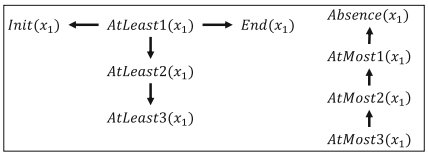
- Behavioral signals within the same cluster ($clust_i$) exhibit high similarity based on a chosen similarity metric.
- Behavioral signals in different clusters ($clust_i$ and $clust_j$, $i \neq j$) exhibit distinct behavioral characteristics.

The choice of clustering algorithm and similarity metric is arbitrary and can be tailored to the specific requirements of the analysis [3].

In our proposed framework, we use agglomerative hierarchical clustering due to its flexibility and interpretability [24]. Agglomerative clustering starts with each behavioral signal $s_c \in \mathcal{S}'_L$ as its own cluster and iteratively merges the two



(a) Subsumption relationships between declarative templates involving two variables. More restrictive templates, such as $ChainSuccession(x_1, x_2)$, subsume less restrictive ones, such as $Succession(x_1, x_2)$.



(b) Subsumption relationships between declarative templates involving a single variable. For example, $AtMost1(x_1)$ is more restrictive than $AtMost2(x_1)$.

Fig. 6 Illustration of subsumption relationships among declarative templates

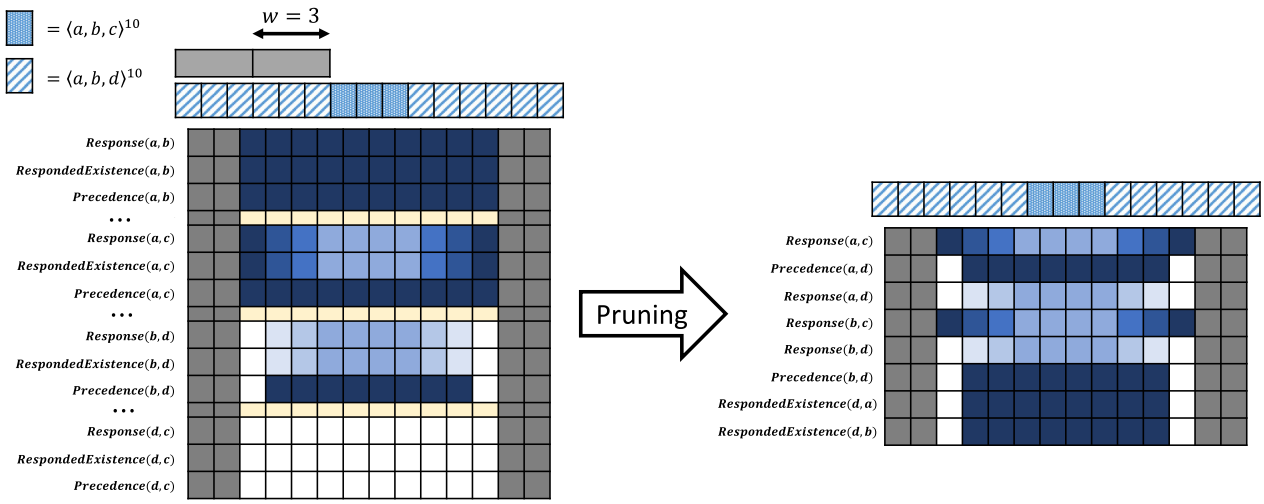


Fig. 7 Pruning step of the feature generation framework illustrated using event log L_3. Blue color intensity reflects the confidence of behavioral signals across time windows. Constant and subsumed signals are removed, yielding a more compact and informative feature set

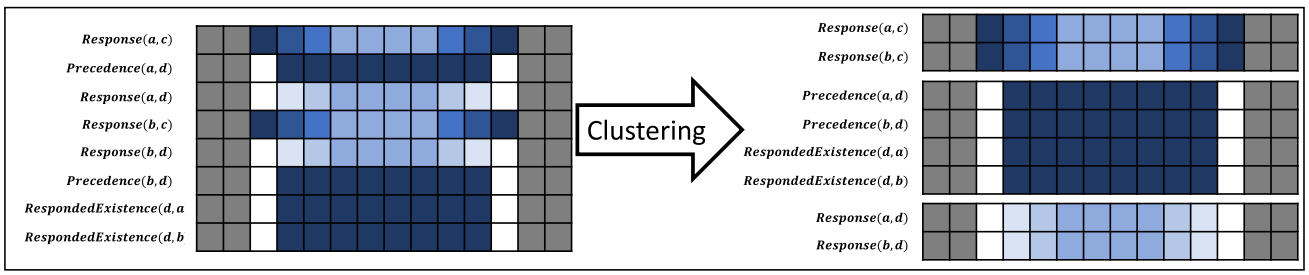


Fig. 8 Clustering of behavioral signals based on their behavioral signal patterns. Features with similar dynamics across windows are grouped together, revealing sets of behavioral signals that evolve in tandem

most similar clusters based on a chosen similarity metric. This process continues until all signals are merged into a single cluster. The similarity between clusters can be measured using distance metrics, e.g., Euclidean distance, and the linkage criteria, e.g., single linkage, complete linkage, or average linkage, determine how distances between clusters are calculated. A key advantage of agglomerative hierarchical clustering is that it does not require the number of clusters to be predefined. Instead, it produces a hierarchical structure that organizes signals into clusters at various levels of granularity, allowing the identification of sets of constraints with similar behavioral patterns.

4.2.4 Running example

In real-world scenarios, processes often involve many activities, and the number of generated features can be large. In our running example, $\mathcal{S}_{L_6} = declset(L_6)$ includes 384 declarative constraints, resulting in 384 behavioral signals in $\mathcal{S}_{L_6}$. During the pruning step:

- 240 signals are removed for being always zero,
- 20 signals are removed for being always one,
- 80 signals are removed due to hierarchical relations among declarative constraints.

After pruning, the set of behavioral signals is reduced to $\mathcal{S}'_{L_6} = \text{prune}(\mathcal{S}_{L_6})$, containing 74 signals retained for further analysis. These remaining signals are then clustered using hierarchical agglomerative clustering. The resulting hierarchical structure can be visualized through a dendrogram, which illustrates how clusters are formed at different levels of similarity. Figure 9 presents this dendrogram, enabling the selection of an appropriate number of clusters either visually, by applying a predefined distance threshold, or through automated selection methods.

The suitable number of clusters, determined to be 7, reveals distinct behavioral patterns. In Fig. 10, the clustered behavioral signals are illustrated, with each cluster grouping related signals. The color intensity for each behavioral signal corresponds to the confidence level of the associated declarative constraint within a given window. Vertical dashed lines show the boundaries between clusters, while red horizontal lines indicate the change points detected using the earth mover's distance framework.

4.3 Integration of segments and clusters for global insights

After segmentation via the EMD measure, we perform a pairwise comparison of the segments and correlate the identified segments with the most significant behavioral cluster, enabling a deeper understanding of the control-flow characteristics underlying each segment. This novel combination of segmentation, feature analysis, and correlation provides process experts with both precise detection of change points and meaningful insights into the process behaviors driving these changes.

The proposed framework identifies control-flow change points and behavioral clusters to provide localized and aggregated insights into process behavior. To derive a more comprehensive understanding, we integrate these perspectives through three steps. First, we perform segment-to-segment comparisons to analyze how process behavior evolves between adjacent and non-adjacent segments. Next, we conduct segment-to-cluster comparisons to uncover the relationships between specific segments and clusters, using correlation analysis to quantify these associations. Finally, we refine clusters by identifying and ranking the most informative declarative constraints, providing interpretable explanations of the behavioral characteristics for each segment.

4.3.1 Segment-to-segment comparison

Based on the visualizations in Fig. 4, control-flow behavior may exhibit multiple changes across the range of the indicator. However, for each detected change, the framework only identifies that w buckets on the left differ from w buckets on the right. To provide more comprehensive global insights, the framework is extended to segment the event log based on peaks in the $ldist$ values. Using specified b and w parameters, each peak in $ldist$ marks a segmentation point, dividing the event log into segments. Each segment comprises cases from the previous change point to the current change point. This segmentation enables not only the analysis of adjacent segments but also the comparison of non-adjacent segments using the earth mover's distance measure, facilitating the identification of broader patterns in process behavior.

Definition 16 *(Change Point in Control Flow)* Let $L \in \mathcal{L}$ be an event log, κ be a case-level indicator, $b \in \mathbb{N}^{[2,|L|]}$ be the number of buckets specified by the user, and $w \in \mathbb{N}^{[1,\frac{b}{2}]}$ be a window size parameter. Additionally, let $\theta_{cp} \in [0, 1]$ be a user-defined threshold to evaluate the significance of $ldist$. A point $p \in [w, b - w]$ is a change point in the control-flow behavior if the following conditions are met:

- $ldist(p) \geq \theta_{cp}$
- $ldist(p-1) \leq ldist(p)$, if $p \in (w, b-w]$
- $ldist(p+1) \leq ldist(p)$, if $p \in [w, b-w)$

Definition 17 *(Segments and Segment Signals)* Let $P = \langle p_1, \ldots, p_{|P|} \rangle$ be the ordered sequence of change points in the $ldist$ function, such that $\forall i, j \in \{1, \ldots, |P|\}$, $p_i \leq p_j$ if and only if $i < j$. Given $|P|$, the number of peaks in the

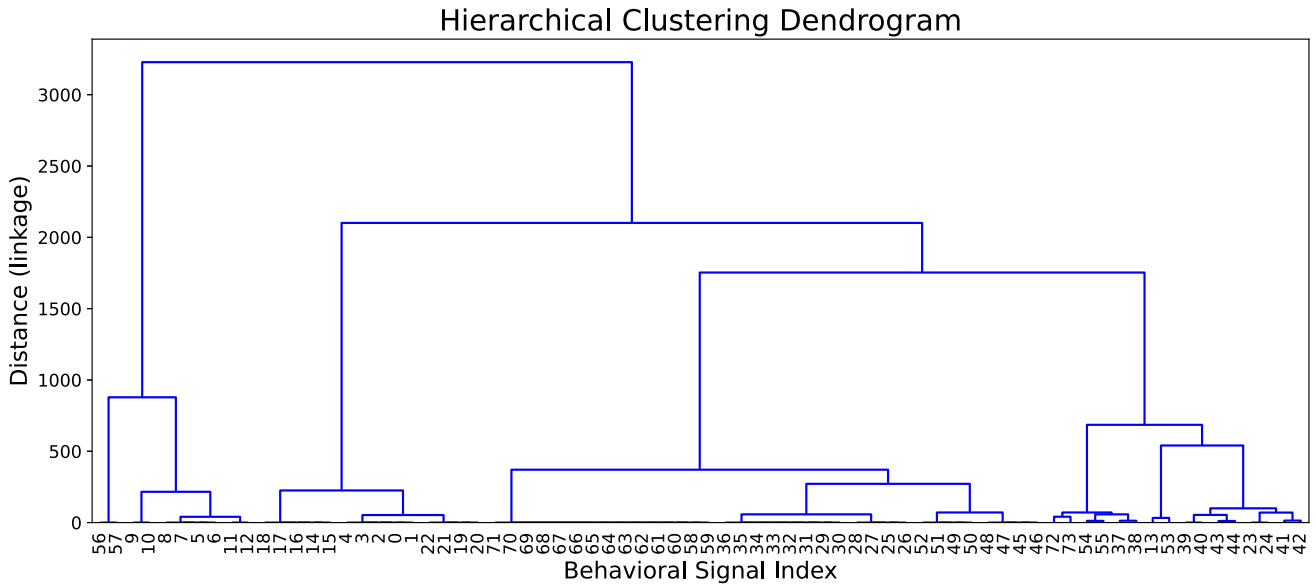


Fig. 9 Hierarchical clustering dendrogram of the pruned behavioral signals ($\mathcal{S}'_{L_6}$). The tree structure reflects merging behavior at different similarity levels and can be used to define an appropriate number of clusters

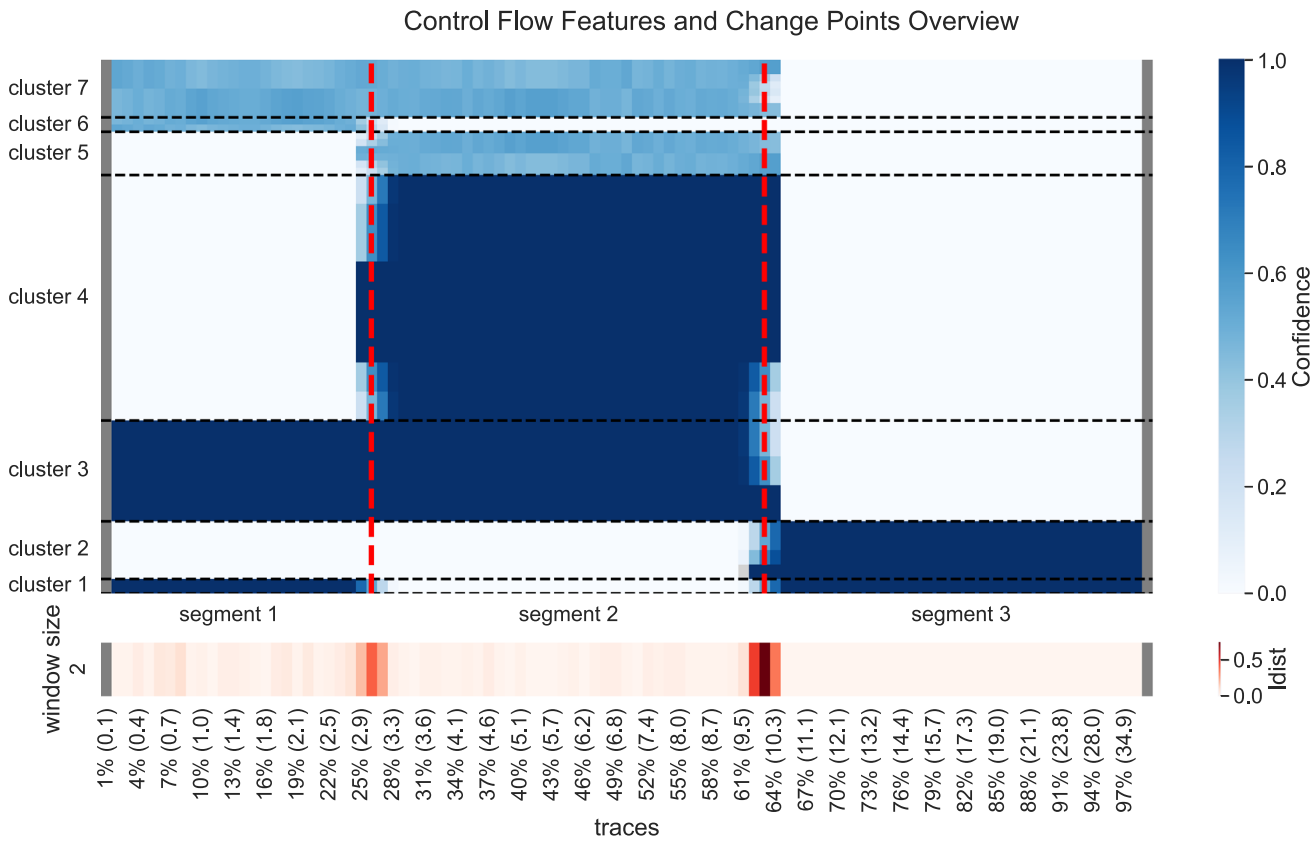


Fig. 10 Control-flow feature clusters derived from behavioral signals and change point detection. The red dashed lines are the change points detected using the EMD-based framework considering $\theta_{cp} = 0.1$

$ldist$ function, the event log is divided into $|P|+1$ segments, denoted as $seg_1, seg_2, \ldots, seg_{|P|+1}$, defined as follows:

- $seg_1 = \{\sigma \in B_x \mid x \leq p_1\}$,
- $seg_i = \{\sigma \in B_x \mid p_{i-1} < x \leq p_i\}, \quad \forall i \in [2, |P|]$,
- $seg_{|P|+1} = \{\sigma \in B_x \mid p_{|P|} < x\}$.

Additionally, considering $b \in \mathbb{N}^{[2,|L|]}$ as the number of buckets and $w \in \mathbb{N}^{[1,\frac{b}{2}]}$ as the window size parameter . For each segment seg_i, define its corresponding segment signal $sig_i = \langle z_w, \ldots, z_{b-w} \rangle$, where $k \in [w, b-w]$ and:

- $z_k = 1$, if $k \leq p_1$ for $i = 1$,
- $z_k = 1$, if $p_{i-1} < k \leq p_i \quad \forall i \in [2, |P|]$,
- $z_k = 1$, if $p_{|P|} < k$ for $i = |P| + 1$,
- $z_k = 0$, otherwise.

To further enhance the analysis, we hierarchically merge similar segments when the control-flow behavior is not significantly different, as determined by a user-defined difference threshold $\theta_{cp} \in [0, 1]$. This step aims to simplify the segmentation by consolidating segments with minimal variation. The merging process is performed recursively: At each step, the two segments with the smallest $ldist$ value are merged if their distance is below the threshold θ_{cp}. This ensures that only meaningful and significant changes in behavior are preserved, while noise or minor variations are excluded from the analysis.

4.3.2 Segment-to-cluster correlation

To gain insights into how process behavior evolves, we analyze the relationship between segments and control-flow behavioral clusters. To quantify these relationships, we perform a correlation analysis that measures the association between each segment and the identified clusters. The strength of this relationship indicates which clusters are most relevant for explaining the behavioral dynamics in each segment.

Definition 18 *(Correlation Between Behavioral Clusters and Segments)* Let $corr : \mathbb{R}^* \times \mathbb{R}^* \to [-1, 1]$ denote the Pearson correlation coefficient between two signals. For a segment signal sig_i and a behavioral cluster $clust_j$ where $i \in \{1, \ldots, |P| + 1\}$, with $|P| + 1$ being the total number of segments generated from $|P|$ change points, and $j \in \{1, \ldots, n\}$, with n being the total number of clusters. The correlation between the segment and the cluster is defined as:

$$corr(sig_i, clust_j) = \frac{\sum_{s_c \in clust_j} corr(sig_i, s_c)}{|clust_j|},$$

This measure quantifies the average correlation between the segment signal and the behavioral signals within the cluster.

Finally, we visualize the results of the correlation analysis using a heatmap, which illustrates the strength of the relationships between segments and clusters. This visual representation enables process experts to quickly identify the most relevant clusters for each segment and gain a deeper understanding of the behavioral patterns that characterize the process over the indicator range.

4.3.3 Global insights generation

While the pruning step eliminates non-informative behavioral signals, the resulting clusters of constraints can still be large, making them difficult to interpret. To enhance clarity and provide better insights, a post-processing step is performed on each cluster to remove redundant constraints and rank the remaining ones based on specific criteria. For instance, if $AtLeast1(a)$ and $RespondedExistence(a, b)$ appear in the same cluster, $CoExistence(a, b)$ can be considered redundant. This redundancy arises because $AtLeast1(a)$ ensures a must always occur, and $RespondedExistence(a, b)$ guarantees that whenever a occurs, b does as well. The approach proposed in [13] is utilized to systematically identify and remove such redundant constraints.

In conjunction with the correlation analysis between segments and clusters, the ranking of constraints is tailored to the segment being analyzed. This ranking considers whether positive or negative correlations are of greater interest, depending on the specific behavioral characteristics of the segment that require explanation. By refining the clusters and prioritizing the most relevant constraints, this step provides a concise and interpretable summary of the process dynamics for each segment.

Definition 19 *(Behavioral Signal Scores in Clusters)* Let sig_i represent the segment signal corresponding to segment i, and s_c be a behavioral signal within a cluster $clust_j$, where $i \in \{1, \ldots, |P| + 1\}$, with $|P| + 1$ being the total number of segments generated from $|P|$ change points, and $j \in \{1, \ldots, n\}$, with n being the total number of clusters. To rank the constraints within a cluster $clust_j$ based on their positive or negative correlation with segment i, we define the following *score*:

$$score_{clust_j, sig_i}(s_c) = \frac{s_c \cdot sig_i}{sig_i \cdot sig_i} - \frac{\sum_{k \neq i} s_c \cdot sig_k}{\sum_{k \neq i} sig_k \cdot sig_k}$$

where $s_c \cdot sig_i$ represents the dot product of the signals s_c and sig_i, and $sig_i \cdot sig_i$ represents the self-dot product (norm squared) of sig_i.

This score is utilized to rank constraints within a cluster. It quantifies the difference between the average behavioral signal value in the specified segment and in other segments. A score of $+1$ indicates that the signal exclusively takes a value of one within the specified segment, while a score of -1 signifies that the signal consistently takes a value of one in all other segments except the specified segment.

Declarative constraints are inherently interpretable in natural language, which facilitates effective communication with domain experts. To enhance readability and accessibility, this paper adopts textual representations of declarative constraints, providing clear and intuitive explanations of the underlying process behavior.

4.3.4 Running example

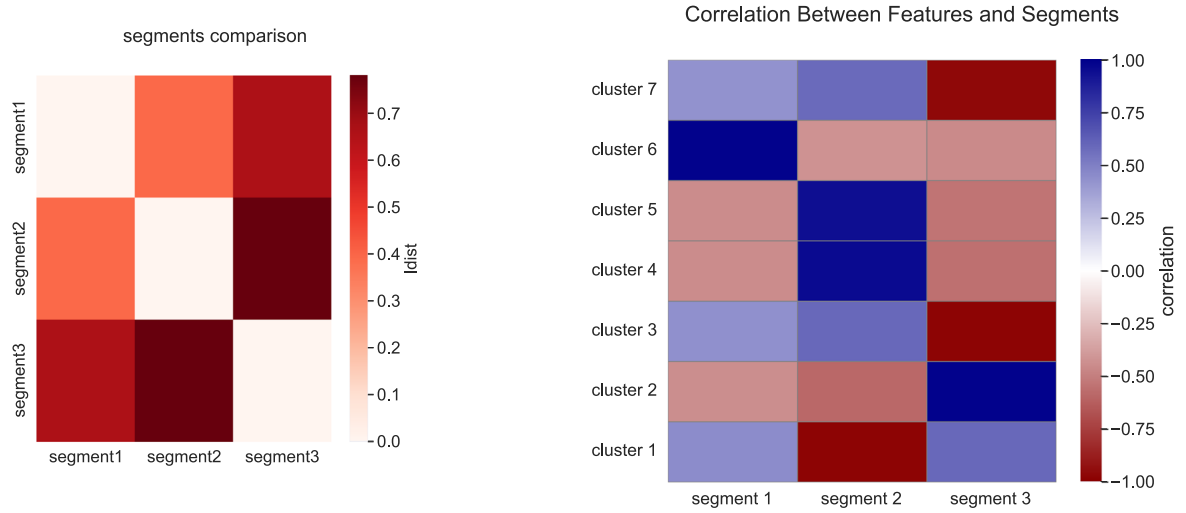
Figure 11b visualizes the correlation between process segments and behavioral clusters. The heatmap highlights clusters with strong positive or negative correlations to each segment, offering insights into the relationships between segments and their corresponding control-flow behaviors. This visualization reveals how specific clusters align with changes at distinct points, emphasizing the behavioral characteristics that shift between segments. Positive correlations are shown in blue and negative correlations in red, with darker shades representing higher absolute correlation. Based on this heatmap:

- For segment 1, Cluster 6 with $corr(sig_1, clust_6) = 0.98$ shows a high positive correlation.
- For segment 2, Clusters 4 with $corr(sig_2, clust_4) = 0.95$ and 5 with $corr(sig_2, clust_5) = 0.94$ have high positive correlations, while Cluster 1 with $corr(sig_2, clust_1) = -0.98$ shows a high negative correlation.
- For segment 3, Cluster 2 with $corr(sig_3, clust_2) = 0.99$ has a high positive correlation, while Clusters 3 with $corr(sig_3, clust_3) = -0.97$ and Cluster 7 with $corr(sig_3, clust_7) = -0.96$ exhibit high negative correlations.

Comparing these correlation scores with the visual intuitions from Fig. 5 demonstrates the effectiveness of this heatmap in uncovering meaningful relationships between clusters and signals, reinforcing the explanatory power of the framework. Below we summarize the key declarative constraints for clusters 1 to 4 based on the scores calculated using Def. 19.

- Cluster 1 (includes 2 behavioral signals, sorted based on $-score_{clust_1,sig_2}$):
  - *in-person interview 2* must never occur (*score*: -95.36, rank 1).
  - *request documents* must never occur (*score*: -95.36, rank 2).
- Cluster 2 (includes 6 behavioral signals, sorted based on $score_{clust_2,sig_3}$):
  - *create application* and *cancel application* occurs if and only if *cancel application* immediately follows *create application* (*score*: 96.87, rank 1).
  - *cancel application* is the last to occur (*score*: 96.73, rank 3).
  - *check documents* must never occur (*score*: 96.73, rank 4).
  - *in-person interview 1* must never occur (*score*: 96.73, rank 6).
- Cluster 3 (includes 9 behavioral signals, sorted based on $-score_{clust_3,sig_3}$):
  - *cancel application* must never occur (*score*: -96.73, rank 1).
  - *check documents* occurs at least once (*score*: -96.73, rank 2).
  - If *create application* occurs, then *check documents* occurs afterward before *create application* recurs (*score*: -96.73, rank 3).
  - If *create application* occurs, then *in-person interview 1* occurs afterward before *create application* recurs (*score*: -96.73, rank 4).
- Cluster 4 (includes 18 behavioral signals, sorted based on $score_{clust_4,sig_2}$):
  - *create application* and *in-person interview 2* occur if and only if they follow one another, alternating (*score*: 93.88, rank 1).
  - *create application* and *request documents* occur if and only if they follow one another, alternating (*score*: 93.88, rank 2).
  - *in-person interview 1* and *in-person interview 2* occur if and only if they follow one another, alternating (*score*: 93.66, rank 5).
- Cluster 6 (including 2 behavioral signals, sorted based on $score_{cluster_6,sig_1}$):
  - *check documents* is the last to occur (*score*: 49.51, rank 1).
  - *in-person appointment 1* is the last to occur (*score*: 46.58, rank 2).

Figure 1 illustrates distinct behavioral patterns across different risk score ranges serving as the ground truth. In segment 3, which includes cases with a risk score above 10, applications are cancelled immediately after creation. The strong positive correlation of cluster 2 and strong negative correlation of cluster 3 with this segment indicate that the



(a) Pairwise comparison of control-flow behaviors across the three identified segments. The earth mover's distance is used to measure the dissimilarities between segments.

(b) Correlation analysis between identified segments and behavioral feature clusters. The heatmap highlights positive and negative correlations, illustrating how specific clusters explain segment-level behaviors.

Fig. 11 Comprehensive analysis of the motivating example event log using the proposed framework

activity *Create Application* is followed by *Cancel Application*, with no intermediate steps. Trace variants involving other activities appear exclusively in earlier segments. In segment 2, corresponding to risk scores between 3 and 10, we observe a high positive correlation with cluster 4 and a strong negative correlation with cluster 1. This segment uniquely includes activities such as the second interview and document request, which do not occur in other segments.

For segment 1, associated with risk scores below 3, the high correlation with cluster 6 highlights distinctive behavioral signals, such as cases ending with either the first interview or document check. Although correlation analysis reveals key differences, a visual inspection of Fig. 10 provides further insights. For instance, the behavioral signals in cluster 4, related to second interviews and document requests, emerge only after segment 1 and vanish in segment 3. Similarly, the cancellation behavior captured by cluster 3 occurs solely in segment 3, reinforcing the absence of such cases in segments 1 and 2.

5 Evaluation

The proposed framework has been implemented as an open-source tool with a web-based user interface and is publicly available[2]. To evaluate its effectiveness, we conducted different experiments including a case study in collaboration with UWV, the employee insurance agency of the Netherlands, which is responsible for executing social security policies related to unemployment and disability, and two publicly available real-life event logs. UWV provided real-life event data for this study, and we worked closely with domain experts from the agency. Their guidance and feedback ensured that our framework aligns with real-world scenarios and can be effectively applied to practical business contexts. Notably, one of the co-authors of this paper is an employee of UWV and a former PhD researcher in process mining. Their domain expertise significantly shaped the interpretation of the results, and they contributed directly to the discussion and writing of the case study section.

5.1 Data

This section introduces three event logs used to evaluate the applicability of our proposed method: an event log extracted from a claim-handling process of the UWV agency, the road traffic fine management event log, and the BPI challenge 2017 event log. These real-life event logs serve as benchmarks to assess the usability and effectiveness of our approach.

5.1.1 UWV event log:

UWV, the employee insurance agency in the Netherlands, is responsible for managing claims related to unemployment and disability benefits. This study focuses on one of UWV's

[2] https://github.com/aliNorouzifar/X-PVI

core claim-handling processes. A critical area of interest for UWV is understanding how this process evolves with respect to case durations. Specifically, the agency aims to identify change points within case durations, which span from 1 to 476 days.

The event log provided for this analysis contains 144,046 cases, encompassing 484 process variants, 16 distinct activities, and a total of 1,309,719 events. Among all cases, 5,333 (3.7%) are rejected, while 138,671 (96.3%) are accepted, reflecting the primary outcomes of the process. The normative BPMN model of this claim-handling process is depicted in Fig. 12. This process model was manually designed based on an analysis of the event data and discussions with process experts from the agency. The normative process model aligns well with the observed behavior in the event log, achieving an alignment fitness of 99.4% [10].

The process begins with the receipt of a claim from a customer. Ultimately, a claim is either accepted or rejected, represented by the first exclusive choice following the initiation of the claim. A small number of cases (42 claims) neither result in rejection nor acceptance and cannot be adequately explained by the normative model. For rejected cases, the process involves first blocking the claim (*Block Claim 2*) before rejecting it. After rejection, the client may file an objection (*Receive Objection 2*). Only a few cases (68 claims) are accepted eventually after the initial rejection. This behavior is not covered in the normative model.

Another blocking mechanism, *Block Claim 1*, might be applied to a claim. In such cases, the agency requires corrections from the client to unblock the claim and make it ready for acceptance. Once a claim is accepted, subsequent payments are processed. Typically, three payments are made for each accepted claim, with each payment requiring both a payment order and execution. Most cases conclude after this stage. However, in certain instances, additional steps are required. For example, a client might file an objection (*Receive Objection 2*), which could lead to the withdrawal of the claim and repayment of the received benefits. During this process, the claim might be blocked using the *Block Claim 3* mechanism to prevent any further payments.

5.1.2 Road traffic fine management (RTFM):

The RTFM event log documents the administrative handling of traffic fines issued by an Italian municipality. This event log contains 150,370 traces and 150,370 events, covering 231 distinct trace variants and 11 unique activities. It captures the full lifecycle of each fine, from creation to payment or credit collection, including activities such as fine issuance, notification dispatch, penalty application, and the processing of payments or appeals. Each case corresponds to a traffic violation, with events detailing interactions between involved entities, such as citizens and administrative offices. Given that such fine collection processes are typically governed by strict deadlines and time-based administrative regulations, proper handling is essential for ensuring timely payments and enforcing penalties for non-compliance. This context motivates the hypothesis that analyzing the development of control flow in relation to case duration may reveal meaningful behavioral patterns.

5.1.3 BPI challenge 2017 (BPIC17)

The BPIC17 event log captures a comprehensive record of a loan application process from a Dutch financial institution. This event log encompasses 31,509 loan applications and 433,444 events related to the application and offer subprocesses, involving 18 distinct activities and 2,630 unique trace variants. Each application may involve multiple offers, reflecting the institution's policy of providing various loan options to applicants. The event log details the progression of each application through various stages, including submission, assessment, offer creation, validation, and final decisions such as approval, cancellation, or denial. The goal for choosing this event log is to investigate if the process involves some duration based decision-making influencing the control flow of the process.

5.2 Setup

Since our proposed method offers a unique perspective by analyzing control-flow developments through performance indicators, the resulting insights are not directly obtainable using existing approaches. However, with some adaptations, some concept drift detection methods can be employed to extract insights for comparison purposes.

5.2.1 The choice of baseline

The visual drift detection method [34] and the explainable concept drift framework [2] are among the state-of-the-art techniques for identifying concept drift over time. While these approaches are not designed to directly capture performance-based process variability, they can be adapted for such purposes. The key adaptation involves ordering the traces according to the chosen performance indicator (e.g., case duration), and shifting the sliding window accordingly, thus replacing time-based segmentation with performance-based segmentation.

The framework proposed in [2] extracts a comprehensive set of numerical features that capture various process perspectives such as control flow, performance, and resource utilization, and transforms each feature into an individual time series using sliding window segmentation. Next, the method applies change point detection algorithms to every series, flagging statistically significant shifts that sig-

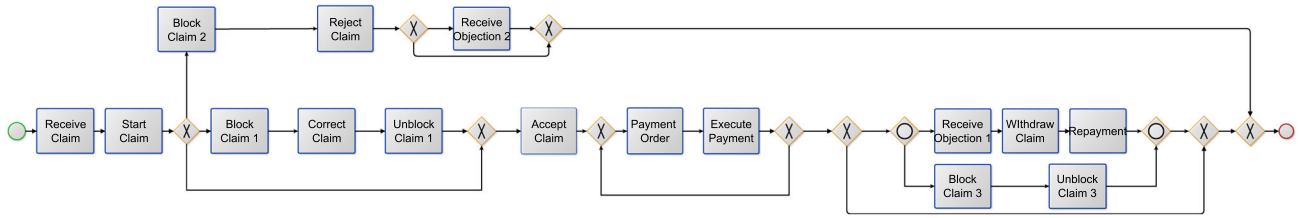


Fig. 12 Normative BPMN model representing the investigated UWV claim-handling process

nal potential concept drifts. To uncover causal relationships between these concept drifts, the authors employ a kernel-based Granger causality test, enabling the detection of both linear and nonlinear cause-and-effect links. The final output is therefore not merely a list of drift timestamps but an explainable set of drift pairs that indicate which feature changes are statistically likely to drive others, giving analysts concrete clues about the underlying reasons for each process change.

We did not consider this framework as our baseline because it provides a fundamentally different form of explainability. Specifically, it investigates causal relationships between change points in *individual* time–series features and can, for instance, conclude that a shift in behavioral signal s triggered a shift in behavioral signal s'. In contrast, our analysis adopts a **global** perspective: we first detect change points for the *entire* process by jointly considering all behavioral signals, and only then determine which clusters of signals exhibit altered characteristics at each detected change point.

The approach introduced in [34] uses the behavioral signals as input and employs the pruned exact linear time (PELT) algorithm in order to identify the change points. PELT is designed to identify significant changes in the mean, variance, or other statistical properties of a sequence of data points without requiring prior knowledge of the number of change points. This flexibility makes it particularly suitable for our use case, where the number of behavioral shifts in the process control flow is unknown. In contrast, the change points identified in our framework are not based on the behavioral signals as a feature space, instead, we use an EMD-based method to identify the changes in the control flow. We compare the visual outputs of this method with the proposed method in our paper. In order to be able to compare the results with our framework, we made some adaptations. The original approach sorts the traces based on time, and we changed it to the performance indicator. The algorithm has window size and sliding parameter, to have similar scale as our framework, and we chose the window size of $w \times \frac{|L|}{b}$ and sliding steps as $\frac{|L|}{b}$.

Note that in the results reported for the baseline method, confidence and support are defined at the trace level. However, in our framework, we adopt event-level measurements instead, as they offer more fine-grained and interpretable insights. Furthermore, the baseline method does not include a coverage-based filtering mechanism, but in our framework, we incorporated our own filtering strategy to eliminate noisy patterns and ensure that the extracted rules reflect meaningful behavioral patterns. The clustering criterion in the baseline method is based on a fixed distance threshold that has not been changed in different experiments.

Table 4 Summary of parameter settings used across all experiments

Event Log	θ_{cp}	w	$\#cl$	θ_{cvg}
UWV	0.10	2	5	0.02
UWV only rejected	0.10	2	7	0.02
UWV only accepted	0.10	2	4	0.02
RTFM	0.14	2	7	0.05
BPIC17	0.20	2	8	0.05

5.2.2 Parameter settings

In all experiments, we set $b = 100$ to enable intuitive, percentage-based insights, i.e., each bucket represents 1% of the cases, sorted by a performance indicator. To evaluate the impact of different window sizes, we test values of 2, 5, 10, and 15. In general, smaller window sizes are preferred for finer granularity, unless the resulting patterns are too noisy, in which case larger windows help smooth the trends. The threshold θ_{cp} is chosen per experiment to ensure that the identified segments align with visual expectations based on observed changes in the $ldist$ values. The other parameters value may vary depending on the characteristics of each experiment. Table. 4 summarizes the parameter settings for each individual experiment.

The parameter θ_{cvg} is chosen based on the characteristics of each experiments to suppress noisy patterns in the behavioral signals. This threshold prevents the influence of constraints that are activated only a few times but exhibit high confidence, which is an artifact of the fact that constraint confidence does not consider the frequency of activation. Table. 5 shows for each experiment, how many behavioral signals are generated, the number of removed constraints due to the constant signal value in all windows and the num-

Table 5 Summary of behavioral signal pruning across experiments, including the number of extracted signals (# extracted), those removed due to constant values (# constant), removed due to subsumption (# subsumed), and the final number of signals retained for analysis (# remaining)

Event log	# Extracted	# Constant	# Subsumped	# Remaining
UWV	2,784	2,206	94	484
UWV only accepted	2,784	2,511	65	228
UWV only rejected	2,445	2,008	63	374
RTFM	1,309	1,031	94	184
BPIC17	3,528	2,102	391	1,035

ber of removed signals due to the hierarchical subsumption relationship to other signals. The last column illustrates the number of remaining signals at the end that are used for the analysis and the visualizations.

The number of clusters for each experiment is determined such that it provides the best visual interpretability. Specifically, we analyze the dendrograms generated through hierarchical clustering as intermediate visualizations to guide the selection of a suitable number of clusters that offer meaningful behavioral distinctions.

5.3 Results

Based on the introduced event logs and experimental setup, we conducted some experiments. In the following, we discuss the insights gained by applying the proposed method for change point detection and explainability extraction.

5.3.1 Case study with UWV (complete event log)

To analyze changes in control-flow behavior over the case duration in the UWV event log, we conducted an experiment using the complete event log. The results are summarized in Fig. 13. Comparative results for control-flow changes, as shown in Fig. 13a, indicate that cases with very short or very long durations exhibit distinct behavior compared to cases in other duration ranges.

Using a threshold $\theta_{cp} = 0.1$ and a window size $w = 2$, the sliding window analysis detected two peaks in the $ldist$ function: one at $i = 3$ (20 days) and another at $i = 95$ (69 days). These peaks divide the event log into three distinct segments, segment 1: [0, 20] days, segment 2: [21, 69] days, and segment 3: [70, 476] days. The pairwise comparisons of these segments using the earth mover's distance are illustrated in Fig. 13c.

Table 6 reports the number of cases and alignment fitness for each segment with respect to the normative BPMN model. The results demonstrate that all segments exhibit high fitness values, indicating strong alignment with the normative process.

To further analyze the segments, the normative BPMN model is highlighted as represented in Fig. 14 to show the process paths associated with each segment. Colors in the model correspond to the frequency of activities in the segments, with transitions appearing in gray if their frequency is below 5%.

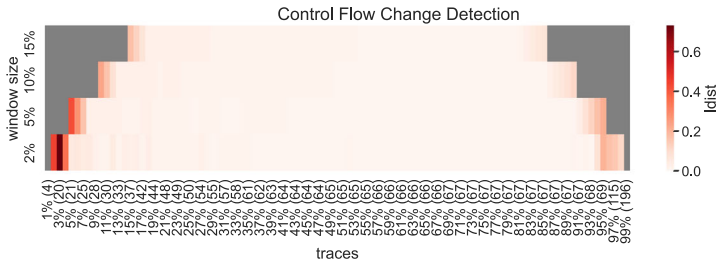
This segmentation and subsequent analysis offer valuable insights into how case durations influence process behavior, providing guidance for enhancing process management. This type of analysis necessitates the availability of a normative model and requires an additional replay of the segmented event logs, which must be appropriately visualized with highlighted activities and arcs. However, if certain behaviors are not captured in the normative model but exhibit significant changes, such behaviors are challenging to visualize and interpret using the normative model alone. Therefore, the explainability extraction step becomes essential, as it broadens the scope of analysis by uncovering a wider range of relationships between activities.

Figure 13b illustrates the generated clusters alongside the change point detection results. Gray points indicate low-coverage values of constraints in specific windows, based on a threshold of $\theta_{cvg} = 0.02$. These points are assigned a value of zero in the behavioral signals. Red dashed lines are the change points detected by the EMD-based technique introduced in this paper.

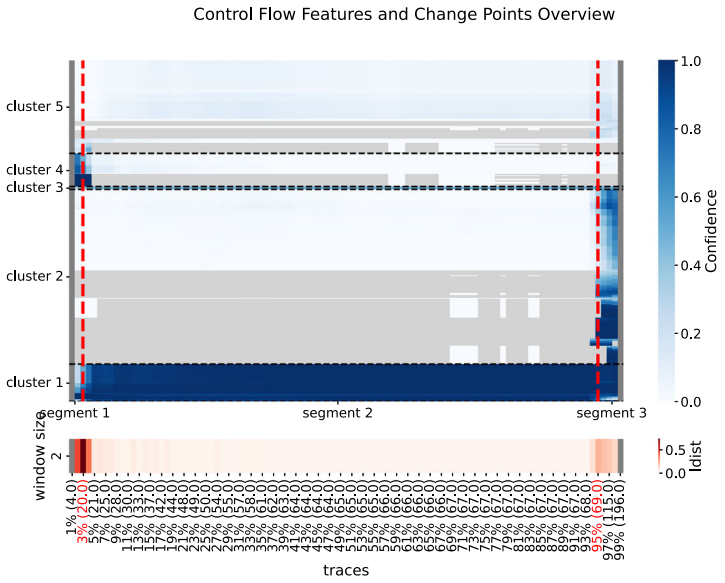
To further analyze the relationship between segments and clusters, a correlation analysis is conducted, as visualized in Fig. 13d. Key observations from the heatmap include:

- **Segment 1**: Displays a high negative correlation with Cluster 3 with $corr(sig_1, clust_3) = -0.89$ and Cluster 1 with $corr(sig_1, clust_1) = -0.65$, while showing a high positive correlation with Cluster 4 with $corr(sig_1, clust_4) = 0.69$.
- **Segment 2**: Exhibits a strong negative correlation with Cluster 2 with $corr(sig_2, clust_2) = -0.80$.
- **Segment 3**: Shows high positive correlations with Cluster 2 with $corr(sig_3, clust_2) = 0.91$ and Cluster 5 with $corr(sig_3, clust_5) = 0.57$.

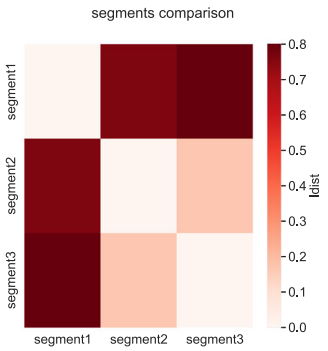
The score introduced in Def. 19 is used in Table 7 to highlight and rank the most relevant behavioral signals within each cluster that contribute to explaining the targeted seg-



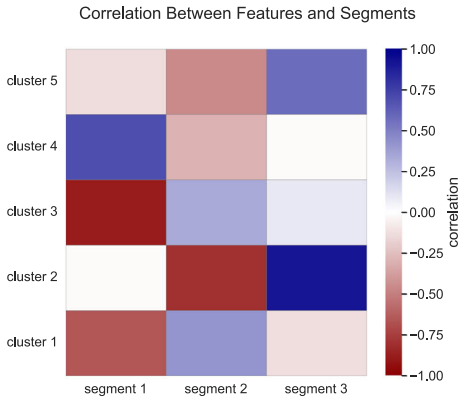
(a) Control flow change detection using the earth mover's distance framework with b=100 and different window sizes $w \in \{2, 5, 10, 15\}$. The color intensity indicates the magnitude of control-flow changes.



(b) Control flow feature clusters derived from behavioral signals and change-point detection. The red dashed lines are the change points detected using the EMD-based framework considering $\theta_{cp} = 0.1$. Gray points correspond to low coverage values ($\theta_{cvg} = 0.02$).



(c) Pairwise comparison of control-flow behaviors across the three identified segments. The earth mover's distance is used to measure the dissimilarities between segments.

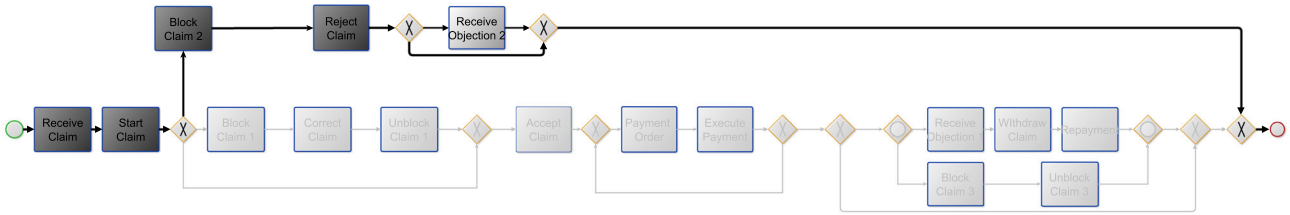


(d) Correlation analysis between identified segments and behavioral feature clusters. The heatmap highlights positive and negative correlations, illustrating how specific clusters explain segment-level behaviors.

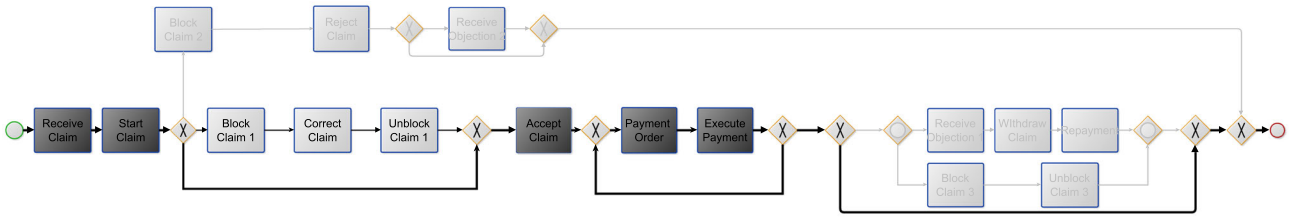
Fig. 13 Comprehensive analysis of the UWV event log using the proposed framework

Table 6 Alignment fitness for segments in the UWV event log, showing conformance to the normative model

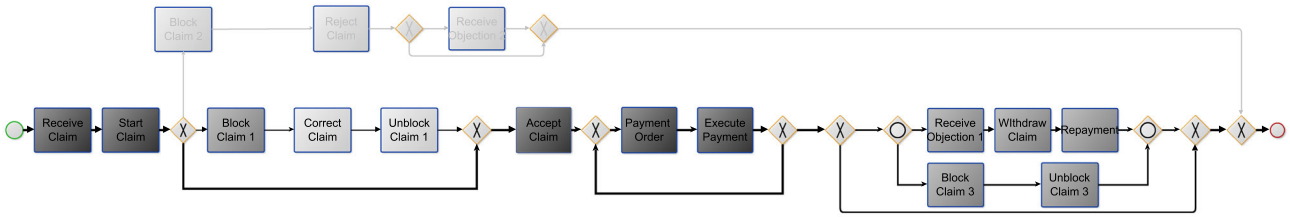
Event log Segment		Number of cases Total	Number of cases Segment	Alignment fitness Total	Alignment fitness Segment
Complete log	Segment 1: [0, 20] days	144,046	4,320	99.4%	99.1%
	Segment 2: [21, 69] days		132,480		99.7%
	Segment 3: [70, 476] days		7,246		94.2%
Rejected claims	Segment 1: [0, 15] days	5,333	3,869	99.3%	99.8%
	Segment 2: [16, 60] days		1,325		98.9%
	Segment 3: [61, 448] days		139		92.3%
Accepted claims	Segment 1: [0, 69] days	138,671	131,765	99.4%	99.7%
	Segment 2: [70, 476] days		6,906		94.0%



(a) Segment 1 in the complete event log with duration in the range [0, 20] days.



(b) Segment 2 in the complete event log with duration in the range [21, 69] days.



(c) Segment 3 in the complete event log with duration in the range [70, 476] days.

Fig. 14 Normative BPMN model is highlighted based on the frequency of transitions replayed for the three segments from the complete event log

ments. Redundant constraints that describe similar behaviors have been excluded[3].

Explainability Through Segment-to-Cluster Associations The segmentation of traces into meaningful intervals provides a foundation for identifying distinct behavioral changes within the process. We explain the behavioral characteristics of each segment and how they relate to representative clusters.

Segment 1: Rejected Claims with Optional Objection Segment 1 primarily includes cases involving activities such as *Receive Claim*, *Start Claim*, *Block Claim 2*, *Reject Claim*, and *Receive Objection 2*. Replay of the cases on the normative process model represented in Fig. 14a suggests that the claims in this segment are typically rejected, although in some cases, the rejection is contested through an objection. This segment correlates strongly with cluster 4, which includes behaviors that are typical for rejected claims, particularly blocking activities that occur before rejection. Conversely, behaviors associated with accepted claims (e.g., payment execution) are mostly absent, as confirmed by the low presence of behavioral signals of cluster 1 in this segment.

[3] The complete list of constraints for all experiments is available at: https://github.com/aliNorouzifar/X-PVI/blob/master/case_study/

Table 7 Top behavioral signals per cluster (sorted by $|score|$), complete event log

Seg	Cl	$\lvert\mathcal{S}'_L\rvert$	Behavioral signal description	$score$	rank
1	1	35	*Payment Order* occurs at least three times	−59.8	1
			Execute Payment occurs at least three times	−59.6	2
			If *Start Claim* occurs, *Payment Order* occurs as well	−59.4	3
			Accept Claim occurs at least once	−59.4	5
3	2	114	*Block Claim 3* occurs only if preceded by *Payment Order*, with no other *Block Claim 3* in between	98.8	1
			Receive Objection 1 occurs only if preceded by *Payment order*, with no other *Receive Objection 1* in between	95.0	7
			Receive Objection 1 occurs only if *Execute Payment* occurs immediately before it	88.4	9
			Receive Objection 1 and *Withdraw Claim* occur if and only if they follow one another, alternating	84.0	12
			Withdraw Claim occurs if and only if it is followed by *Repayment*	83.2	13
1	3	4	*Receive Claim* and *Execute Payment* occur if and only if they follow one another, alternating	−14.4	1
			Start Claim and *Execute Payment* occur if and only if they follow one another, alternating	−14.5	2
			Receive Claim and *Payment Order* occur if and only if they follow one another, alternating	−14.4	3
			Start Claim and *Payment Order* occur if and only if they follow one another, alternating	−14.4	4
1	4	21	*Block Claim 2* occurs only if preceded by *Receive Claim*, with no other *Block Claim 2* in between	98.0	1
			Block Claim 2 occurs if and only if it is followed by *Reject Claim*	98.0	2
			Block Claim 2 occurs only if *Start Claim* occurs immediately before it	98.0	3
			Execute Payment must never occur	59.4	15
			Payment Order must never occur	59.4	16
			Accept Claim must never occur	59.4	17
			Block Claim 2 occurs at least once	58.7	18

Segment 2: Accepted Claims with Standard Payment Cycle Segment 2 corresponds to a more normative process flow where claims are accepted and processed through a complete payment cycle involving three executed payments (Fig. 14b). This behavior aligns closely with cluster 1, which captures the essential steps of a successful claim resolution. The strong presence and satisfaction of behavioral signals of cluster 1 in segment 2 indicate a high degree of process regularity and efficiency. In contrast, exceptional behaviors such as *Block Claim 3* and claim withdrawal, which are common in cluster 2, or claim rejection, frequently observed in cluster 3 are largely absent in this segment. This further reinforces the characterization of segment 2 as representing the ideal acceptance path within the process.

Segment 3: Accepted Claims with Post-Payment Issues Segment 3 also includes accepted claims, but unlike segment 2, these cases involve deviations such as objections after payments have been made, withdrawals, and repayments of previously received benefits (Fig. 14c). The activity *Block Claim 3* is used to stop further payments during the resolution of these issues. This segment is best explained by cluster 2, which captures extended workflows involving objections and corrections after the standard payment cycle. Overall, segment 3 reflects more complex and longer-running cases that deviate from the ideal process flow and require additional handling.

This cluster-to-segment mapping provides interpretable insights into how process behaviors evolve across different segments. It helps distinguish standard flows from exceptions and reveals the underlying behavioral drivers of performance variation.

Comparison to the Baseline The application of the visual concept drift detection method produces the visualization shown in Fig. 15. While a visual inspection of the diagram suggests the presence of a major change point at the beginning and another reflected across multiple clusters toward the end, the PELT-based change point detection method applied to behavioral signals fails to detect these changes. In contrast, our EMD-based change point detection method successfully identifies them.

A closer examination of the behavioral signals reveals noisy patterns, particularly in the lower portion of the visualization. These artifacts are primarily due to the absence of a coverage filtering mechanism. Specifically, a small number of activations followed by the satisfaction of corresponding constraints give rise to rare patterns that contribute to noise. These noisy signals significantly affect the detection outcome, as the identified change points become overly influenced by such low-frequency behaviors.

5.3.2 Case study with UWV (rejected cases)

The extracted segments are highly correlated with the process outcome, i.e., the rejection or acceptance of a claim. However, the correlation between the extracted segments with specific duration is less clear. Another experiment using the 5,333 rejected cases is performed to get more understanding of the relation between duration ranges and specific process variants. The alignment fitness of the rejected cases is 99.3% with respect to the normative model.

An overview of the results for the rejected cases is presented in Fig. 16. As shown in Fig. 16a, using $w = 2$ and θ_{cp}=0.1, four initial segments are identified with duration ranges of [0,13] days, [14,15] days, [16,60] days, and [61,448] days. Due to the low distance between segments 1 and 2 (less than θ_{cp}=0.1), they are merged into a single segment covering the duration range [0,15] days. This merging is evident in the heatmap in Fig. 16c, where segments 1 and 2 exhibit highly similar colors compared to the other segments.

Table 6 summarizes the number of cases in each segment along with their respective alignment fitness values. The results indicate that segment 3 exhibits lower conformance to the normative model, with an alignment fitness of only 92.3%. Figure 17 provides the normative BPMN model with highlighted portions corresponding to each of the three identified segments.

Figure 17a illustrates the first segment, covering a duration range of [0,15] days. This segment consists of cases that are rejected without frequently receiving objections. It represents customers who file claims despite being aware that they are unlikely to be entitled. The second segment, spanning the duration range [16,60] days, is depicted in Fig. 17b. This segment includes cases that are rejected, with some involving objections from customers. Finally, Fig. 17c illustrates the third segment, encompassing a duration range of [61,448] days. This segment includes claims that were initially rejected but later accepted after objections were filed, as well as cases involving multiple objections or non-standard procedures.

The clustering results, along with change point detection, are summarized in Fig. 16b. Gray points indicate low-coverage values of constraints in specific windows, based on a threshold $\theta_{cvg} = 0.02$. These points were assigned a value of zero in the behavioral signals. The red dashed lines represent the change points identified by the EMD-based algorithm.

To evaluate the relationship between segments and clusters, a correlation analysis is conducted. The heatmap in Fig. 16d highlights the correlations between different segments and clusters.

– **Segment 1**: Demonstrates a strong positive correlation with cluster 1, where $corr(sig_1, clust_1) = 0.95$. It

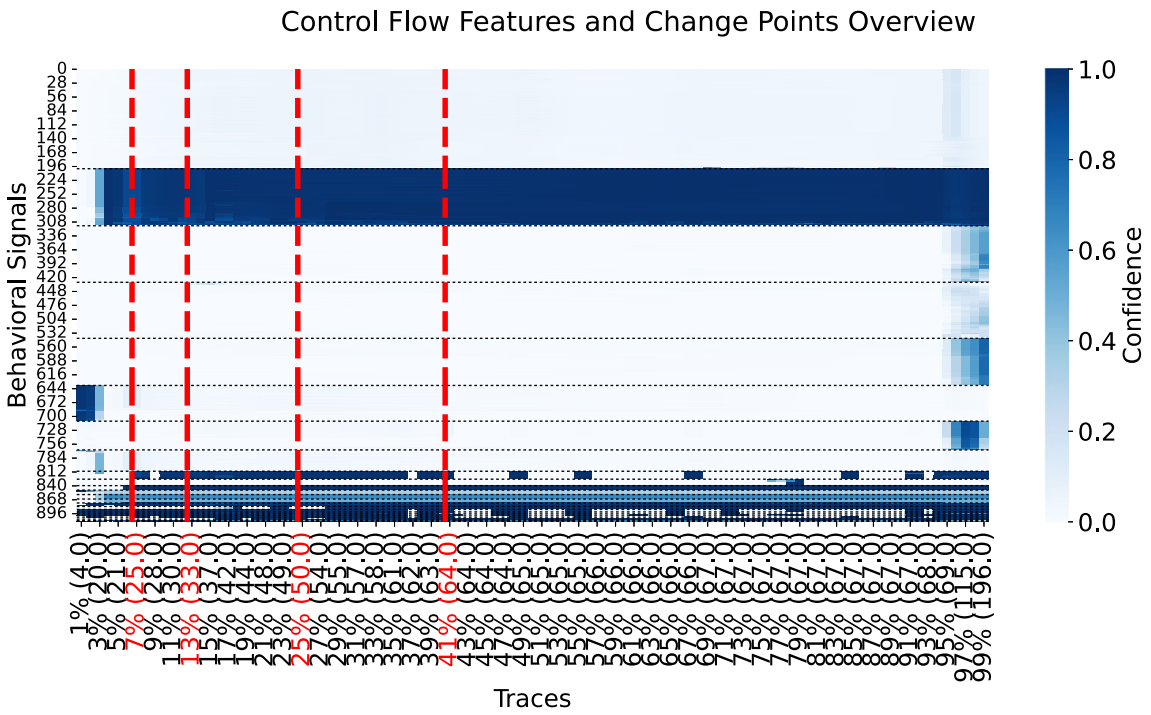


Fig. 15 Visualization of the visual concept drift detection method applied to the UWV event log

also shows strong negative correlations with cluster 3, where $corr(sig_1, clust_3) = -0.73$, and cluster 4, where $corr(sig_1, clust_4) = -0.94$.

- **Segment 2**: Indicates high positive correlations with cluster 3, where $corr(sig_2, clust_3) = 0.71$, and cluster 4, where $corr(sig_2, clust_4) = 0.92$. Additionally, it exhibits a strong negative correlation with cluster 1, where $corr(sig_2, clust_1) = -0.92$.
- **Segment 3**: Displays high positive correlations with cluster 6, where $corr(sig_3, clust_6) = 0.69$, and cluster 7, where $corr(sig_3, clust_7) = 0.59$. It also shows a moderate negative correlation with cluster 2, where $corr(sig_3, clust_2) = -0.64$.

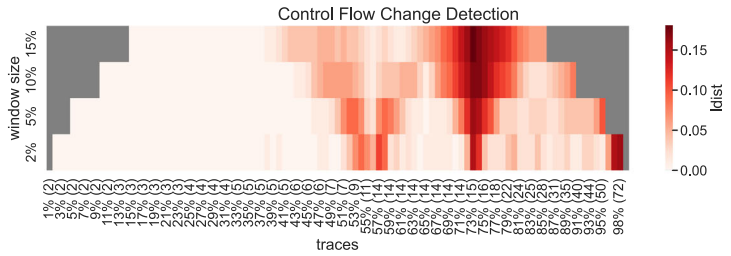
Table 8 provides an analysis of selected constraints for each cluster, emphasizing their alignment with the corresponding segments.

Explainability Through Segment-to-Cluster Associations One of the key strengths of our framework lies in its ability to link changes in control-flow behavior to interpretable clusters of behavioral signals. Each segment, derived through change point detection, exhibits a distinct pattern of behavior that correlates strongly with specific signal clusters.

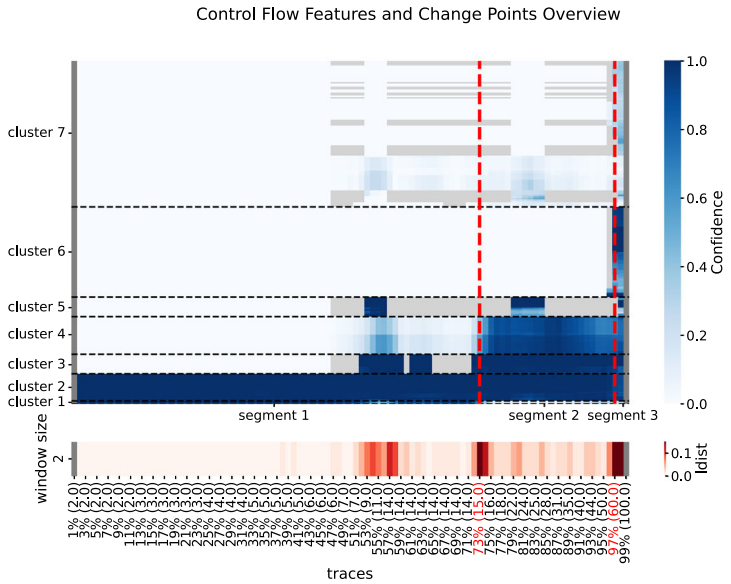
Segment 1: Rejected Claims Without Objections Segment 1 shows a strong positive correlation with cluster 1 and strong negative correlations with cluster 3 and cluster 4. Cluster 1 captures behaviors where *Reject Claim* is the final activity and no *Receive Objection 2* occurs, representing straightforward rejections without appeals. The negative correlation with clusters 3 and 4, both of which involve frequent objection handling, reinforces the interpretation that segment 1 includes cases where the claim is rejected and no follow-up actions are taken within the typical 14-day window. Figure 17a confirms that these cases dominate early in the process.

Segment 2: Emergence of Objections and Follow-ups Segment 2 exhibits a high positive correlation with cluster 3 and cluster 4, and a strong negative correlation with cluster 1. This transition highlights a behavioral shift where rejections increasingly trigger objections. Cluster 3 reflects the frequent presence of *Receive Objection 2*, typically occurring after *Receive Claim* and *Reject Claim*. Cluster 4 also centers on *Receive Objection 2* and its co-occurrence with other rejection-related activities. This indicates that segment 2 comprises more complex cases where rejected claims are actively followed up. The contrast with segment 1 is clearly visible in Figure 17b.

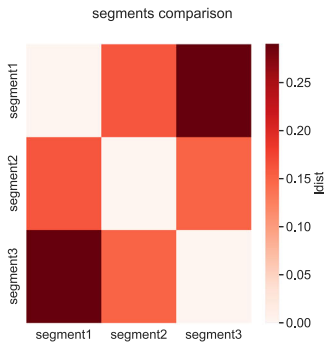
Segment 3: Complex Behavior and Payments After Rejection Segment 3 is characterized by high positive correlation with cluster 6 and cluster 7, and a moderate negative correlation with cluster 2. Cluster 6 captures cases in which rejected claims ultimately result in payments, an exception to the normative process model. This likely involves reconsideration or escalation following an objection. Additionally, cluster 5, which represents unusual uses of *Correct Claim* in rejected cases, becomes more apparent in this segment.



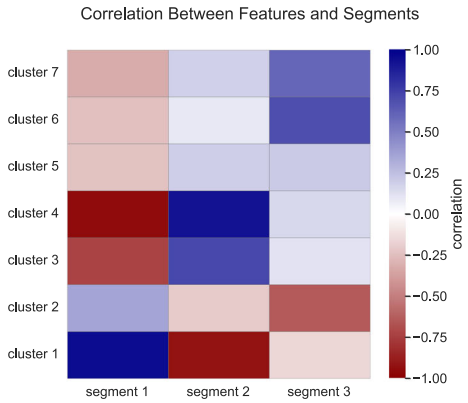
(a) Control flow change detection using the earth mover's distance framework with b=100 and different window sizes $w \in \{2, 5, 10, 15\}$. The color intensity indicates the magnitude of control-flow changes.



(b) Control flow feature clusters derived from behavioral signals and change-point detection. The red dashed lines are the change points detected using the EMD-based framework considering $\theta_{cp} = 0.1$. Gray points correspond to low coverage values ($\theta_{cvg} = 0.02$).



(c) Pairwise comparison of control-flow behaviors across the three identified segments. The earth mover's distance is used to measure the dissimilarities between segments.

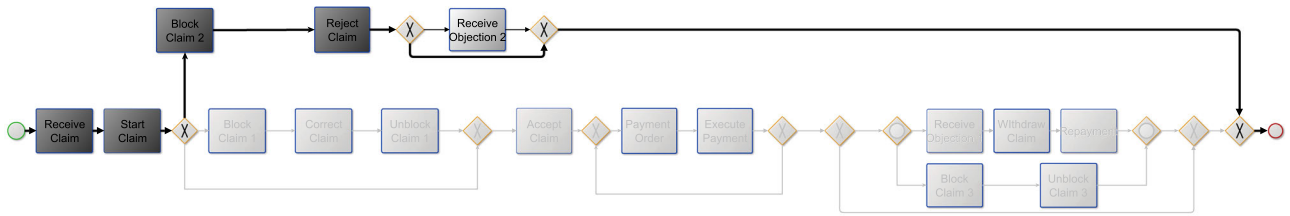


(d) Correlation analysis between identified segments and behavioral feature clusters. The heatmap highlights positive and negative correlations, illustrating how specific clusters explain segment-level behaviors.

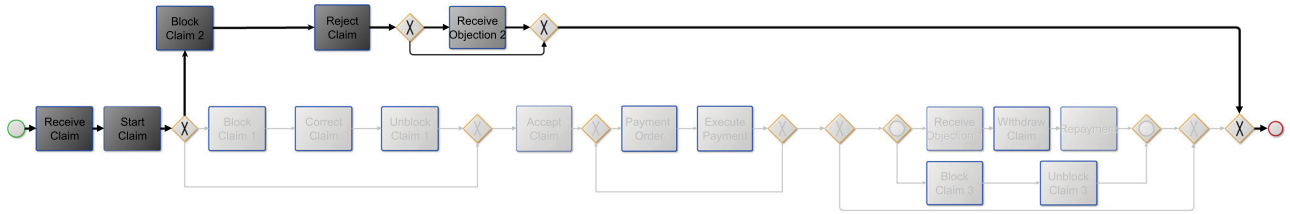
Fig. 16 Comprehensive analysis of the rejected claims in the UWV event log using the proposed framework

Table 8 Top behavioral signals per cluster (sorted by $|score|$), UWV rejected cases

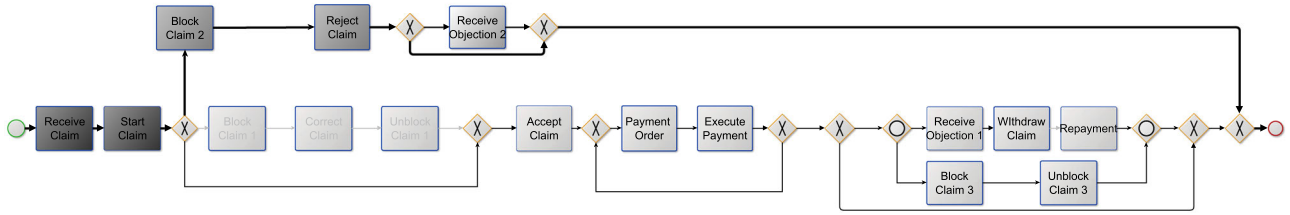
Seg	Cl	$\lvert\mathcal{S}'_L\rvert$	Behavioral signal description	$score$	rank
1	1	2	*Reject Claim* is the last activity to occur	78.3	1
			Receive Objection 2 must never occur	77.1	2
3	2	12	*Receive Objection 2* occurs at most once	−14.5	1
			Reject Claim occurs only if *Block Claim 2* precedes it	−13.0	2
			Accept Claim must never occur	−9.2	5
			Execute Payment must never occur	−9.2	7
1	3	4	*Receive Objection 2* occurs only if *Receive Claim* precedes it	−78.2	1
			Receive Objection 2 occurs only if *Reject Claim* occurs immediately before it	−76.1	4
1	4	16	*Block Claim 2* and *Receive Objection 2* always occur together	−81.8	1
			Reject Claim and *Receive Objection 2* occur only if *Receive Objection 2* immediately follows *Reject Claim*	−78.2	11
1	5	4	*Correct Claim* occurs only if *Receive Claim* precedes it	−9.6	1
			Correct Claim occurs only if *Reject Claim* occurs immediately before it	−6.3	4
3	6	11	*Execute Payment* occurs only if *Payment Order* precedes it	97.9	1
			Execute Payment occurs only if *Reject Claim* precedes it	92.0	5



(a) Segment 1 in the rejected cases event log with duration in the range [0,15] days.



(b) Segment 2 in the rejected cases event log with duration in the range [16,60] days.



(c) Segment 3 in the rejected cases event log with duration in the range [61,448] days.

Fig. 17 Normative BPMN model highlighted based on the frequency of the transitions in replaying the segments from the rejected event log experiment

These behaviors suggest potential deviations or exceptional handling that warrant further investigation.

In summary, each segment reveals a transition in the nature of claim-handling: from direct rejection (segment 1), to the emergence of objections (segment 2), and finally to deviations or reconsidered decisions (segment 3). These interpretable associations between clusters and segments illustrate how our framework supports trace-level reasoning and explainability.

Comparison to the Baseline The heatmap in Fig. 18 illustrates the output of the visual concept drift detection method, which identifies change points based on behavioral signals computed at the trace level. As with Fig. 15, the presence of noisy patterns appears to affect the reliability of the detected change points. While the change points at 15 and 60 days are consistently identified by both methods, the baseline method also detects a change point at 44 days that lacks clear support from visual inspection, suggesting it may be spurious. In contrast, the change point at 14 days appears more plausible, and our EMD-based method, shown in Fig. 16a, confirms its presence when a lower θ_{cp} threshold is applied.

Unlike the PELT algorithm, our approach allows for assessing the relative significance of change points through the visualization of $ldist$ function values. Moreover, the ability to inspect behavioral patterns across different window sizes provides an additional mechanism to assess their robustness: patterns that fade with larger windows are likely noise, while persistent ones may indicate true changes. For instance, the change point identified at 7 days by the baseline method is considered less significant in our approach. When larger window sizes are applied, the corresponding behavioral shift appears less pronounced, though further analysis is needed to draw firm conclusions.

5.3.3 Case study with UWV (accepted cases)

An additional experiment was conducted using 138,671 accepted cases to gain deeper insights into the relationship between case duration ranges and specific process variants. The alignment fitness of the accepted cases with respect to the normative model is 99.4%, indicating a high level of conformance.

The identified segments for the accepted cases are presented in Fig. 19. As illustrated in Fig. 19a, using $w = 2$ and $\theta_{cp} = 0.1$, two distinct segments were detected, corresponding to duration periods of [0,69] days and [70,476] days. A

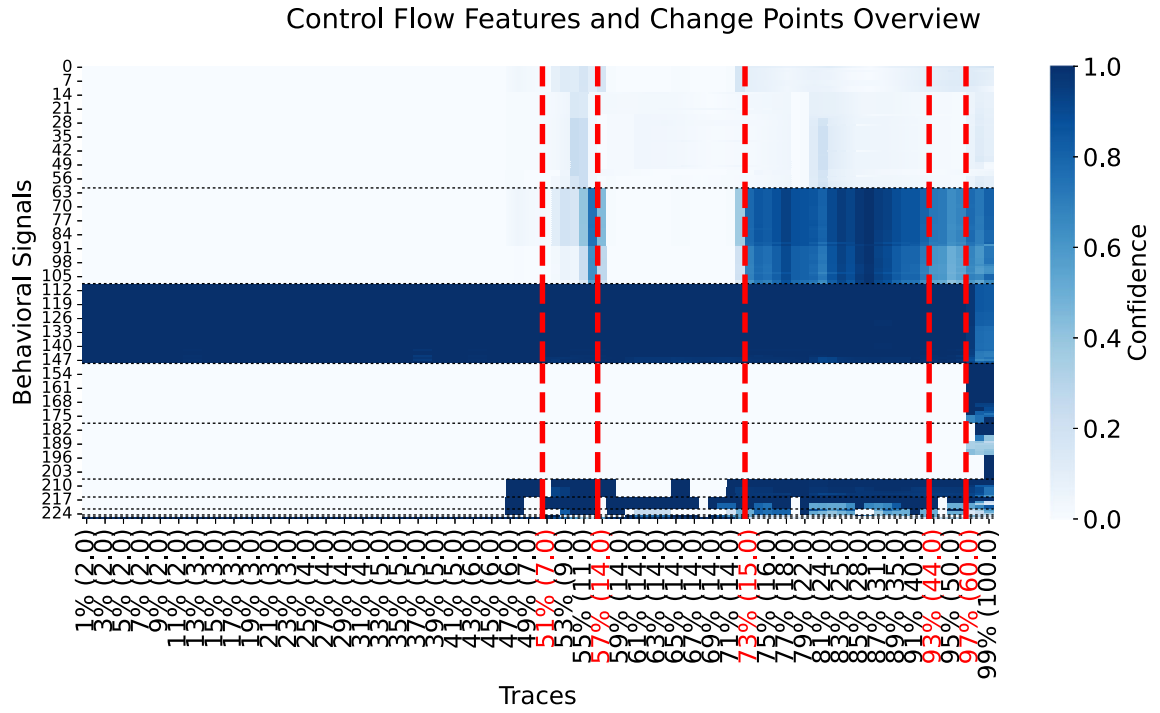


Fig. 18 Visualization of the visual concept drift detection method applied to the rejected cases of the UWV event log

pairwise comparison of these segments, depicted in Fig. 19c, reveals significant differences between the two segments.

The normative models highlighted in Fig. 20 reveal the differences in process behavior between the two segments. The key distinction between *segment 1*, illustrated in Fig. 20a, and *segment 2*, depicted in Fig. 20b, is the occurrence of the subprocess involving *Receive Objection 1*, *Withdraw Claim*, *Repayment*, *Block Claim 3*, and *Unblock Claim 3*. This subprocess is strongly associated with cases that have durations of 70 days or more.

An overview of the behavioral signals is provided in Fig. 19b. Gray points in the graph represent constraints with low-coverage values in specific windows ($\theta_{cvg} = 0.02$), which are assigned a value of zero in the behavioral signals. Red dashed lines indicate change points identified using the EMD-based approach.

The correlation between the segments and clusters is illustrated in Fig. 19d. Key insights from the heatmap include:

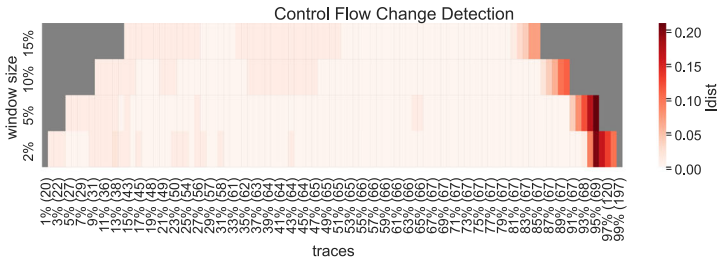
- **Segment 1**: Exhibits a high positive correlation with cluster 1, where $corr(sig_1, clust_1) = 0.80$ and strong negative correlations with cluster 2, where $corr(sig_1, clust_2) = -0.92$, cluster 4, where $corr(sig_1, clust_4) = -0.91$, and cluster 3, where $corr(sig_1, clust_3) = -0.67$.
- **Segment 2**: Shows high positive correlations with cluster 2, where $corr(sig_2, clust_2) = 0.92$, cluster 4, where $corr(sig_2, clust_4) = 0.91$, and cluster 3, where $corr(sig_2, clust_3) = 0.67$, and a strong negative correlation with cluster 1, where $corr(sig_2, clust_1) = -0.80$.

Table 9 reports the calculated *score* for a selection of behavioral signals in specific clusters regarding focused segments.

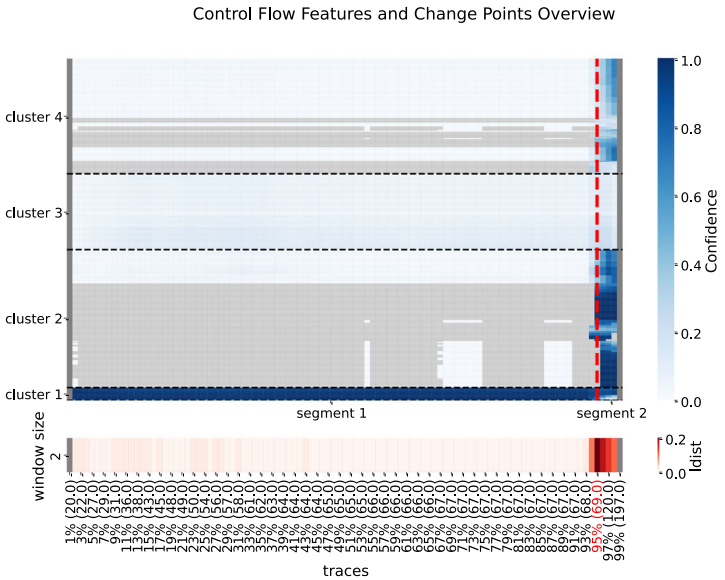
Explainability Through Segment-to-Cluster Associations The segmentation of cases based on performance variation yields meaningful groups of traces that reflect distinct process behaviors. By analyzing the correlation between these segments and clusters of behavioral signals, we gain insight into how specific process dynamics evolve over duration of the cases. Below, we interpret the identified segments by linking them to their most influential clusters.

Segment 1: Efficient Payment Completion Segment 1 is characterized by straightforward case handling where payment execution typically marks the conclusion of the process. This segment shows a strong positive correlation with cluster 1, which contains cases where claims were resolved without any post-payment complications. No actions such as blocking, objections, claim withdrawals, or benefit repayments occur in this group. The negative correlations with clusters 2, 3, and 4 indicate the absence of complex or exceptional behaviors, reinforcing the interpretation that segment 1 reflects efficient and desirable process instances, as shown in Fig. 20a.

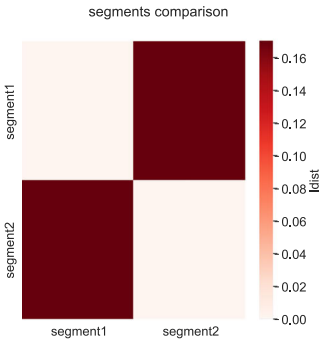
Segment 2: Post-Payment Complications and Rework Segment 2 exhibits high positive correlations with cluster 2,



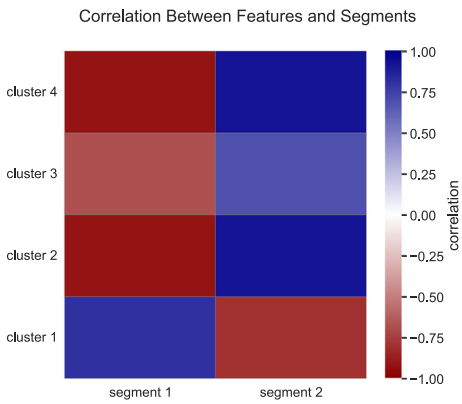
(a) Control flow change detection using the earth mover's distance framework with b=100 and different window sizes $w \in \{2, 5, 10, 15\}$. The color intensity indicates the magnitude of control-flow changes.



(b) Control flow feature clusters derived from behavioral signals and change-point detection. The red dashed lines are the change points detected using the EMD-based framework considering $\theta_{cp} = 0.1$. Gray points correspond to low coverage values ($\theta_{cvg} = 0.02$).



(c) Pairwise comparison of control-flow behaviors across the two identified segments. The earth mover's distance is used to measure the dissimilarities between segments.

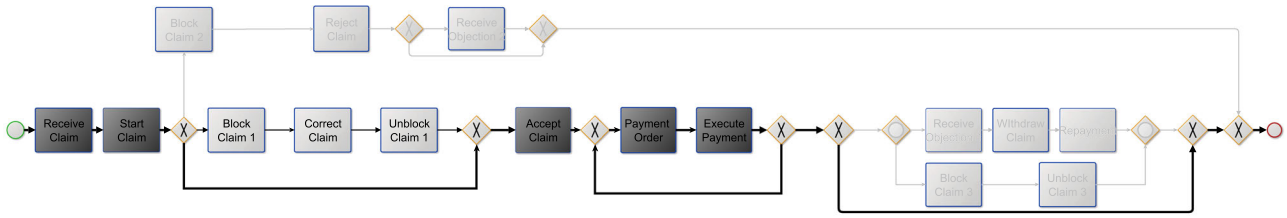


(d) Correlation analysis between identified segments and behavioral feature clusters. The heatmap highlights positive and negative correlations, illustrating how specific clusters explain segment-level behaviors.

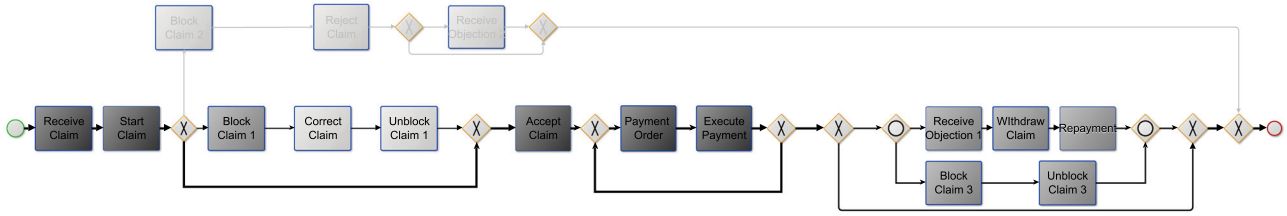
Fig. 19 Comprehensive analysis of the accepted claims in the UWV event log using the proposed framework

Table 9 Top behavioral signals per cluster (sorted by $|score|$), UWV accepted cases

Seg	Cl	$\lvert\mathcal{S}'_L\rvert$	Behavioral signal description	$score$	rank
2	1	13	*Execute Payment* is the last to occur	−68.7	1
			Block Claim 3 must never occur	−60.8	2
			Receive Objection 1 must never occur	−44.1	3
			Withdraw Claim must never occur	−38.0	4
			Repayment must never occur	−28.8	5
2	2	73	*Block Claim 3* occurs only if preceded by *Execute Payment*, with no other *Block Claim 3* in between	98.9	1
			Receive Objection 1 occurs only if *Execute Payment* occurs immediately before it	88.6	2
			Receive Objection 1 and *Withdraw Claim* always occur together	85.6	3
			Withdraw Claim and *Repayment* occur if and only if they follow one another, alternating	82.8	4
			If *Receive Objection 1* occurs, *Block Claim 3* occurs as well	78.3	5
2	4	79	*Receive Objection 1* and *Withdraw Claim* occur if and only if *Withdraw Claim* immediately follows *Receive Objection 1*	61.6	1
			If *Receive Objection 1* occurs, *Repayment* follows	56.7	3
			Accept Claim and *Withdraw Claim* occur if and only if they follow one another, alternating	52.1	8
			Payment Order and *Receive Objection 1* always occur together	49.6	9



(a) Segment 1 in the accepted cases event log with duration in the range [0,69] days.



(b) Segment 2 in the accepted cases event log with duration in the range [70,476] days.

Fig. 20 Normative BPMN model highlighted based on the frequency of the transitions in replaying the segments from the accepted event log experiment

cluster 4, and cluster 3. These clusters represent complex post-payment behaviors, including objections, claim withdrawals, and benefit repayments. Specifically, cluster 2 captures cases that deviate significantly from the ideal payment flow by involving additional steps after initial resolution (Fig. 20b). Cluster 4, while exhibiting similar behaviors, shows slightly weaker confidence in its signals, as reflected in the lighter color intensities in the behavioral heatmap (Fig. 19b). The negative correlation with cluster 1 further confirms that segment 2 diverges from the standard, efficient execution path and instead reflects cases that required rework or correction after payment.

This alignment between segments and behavioral clusters provides interpretable, fine-grained explanations for process variation. Segment 1 can be interpreted as the desirable process outcome, while segment 2 reflects less ideal cases requiring additional resources and interventions.

Comparison to the Baseline Figure 21 presents the results of the visual concept drift detection method. The detected change points, indicated by vertical red dashed lines, appear to be heavily influenced by noisy patterns in the behavioral signals. There is no clear visual correspondence between the identified change points and the observable patterns in the heatmap, raising concerns about their validity.

Although several clusters exhibit signs of behavioral changes in cases with long durations, the PELT method fails to detect these shifts. In contrast, the visualization generated by our approach, shown in Fig. 19b, reveals more interpretable and meaningful change points. This discrepancy likely stems from the influence of low-coverage signals in the baseline method, which may give rise to misleading or spurious patterns.

5.3.4 RTFM

Figure 22 presents the visual insights derived using the proposed method. As illustrated in Fig. 22a, multiple changes in control-flow behavior are consistently observed across all window sizes. Based on this observation, we set the parameters to $w = 2$ and $\theta_{cp} = 0.14$, which led to the identification of three significant change points, leading to the generation of four segments. In Fig. 22b, the output of the change point detection method is combined with the behavioral signal view to highlight the clusters of behavioral signals that contribute to each detected change. Furthermore, Fig. 22c illustrates the pairwise comparison of the identified segments. Based on this visualization, the maximum pairwise distance is observed between segment 1 and segment 4, indicating the highest degree of dissimilarity. In contrast, the relatively lower distance between segment 3 and segment 4 suggests a higher degree of similarity in their control-flow behavior.

In this experiment, the number of clusters for the behavioral signals is set to 7, resulting in distinct groups of signals, each exhibiting patterns associated with the change points identified by the EMD-based method. Figure 22d illustrates the correlation between these clusters and the resulting segments.

- **Segment 1**: This segment shows a strong negative correlation with cluster 2, where $\text{corr}(sig_1, clust_2) = -0.96$, a relatively high positive correlation with cluster 7, where $\text{corr}(sig_1, clust_7) = 0.74$, and a moderately high negative correlation with cluster 3, where $\text{corr}(sig_1, clust_3) = -0.69$.

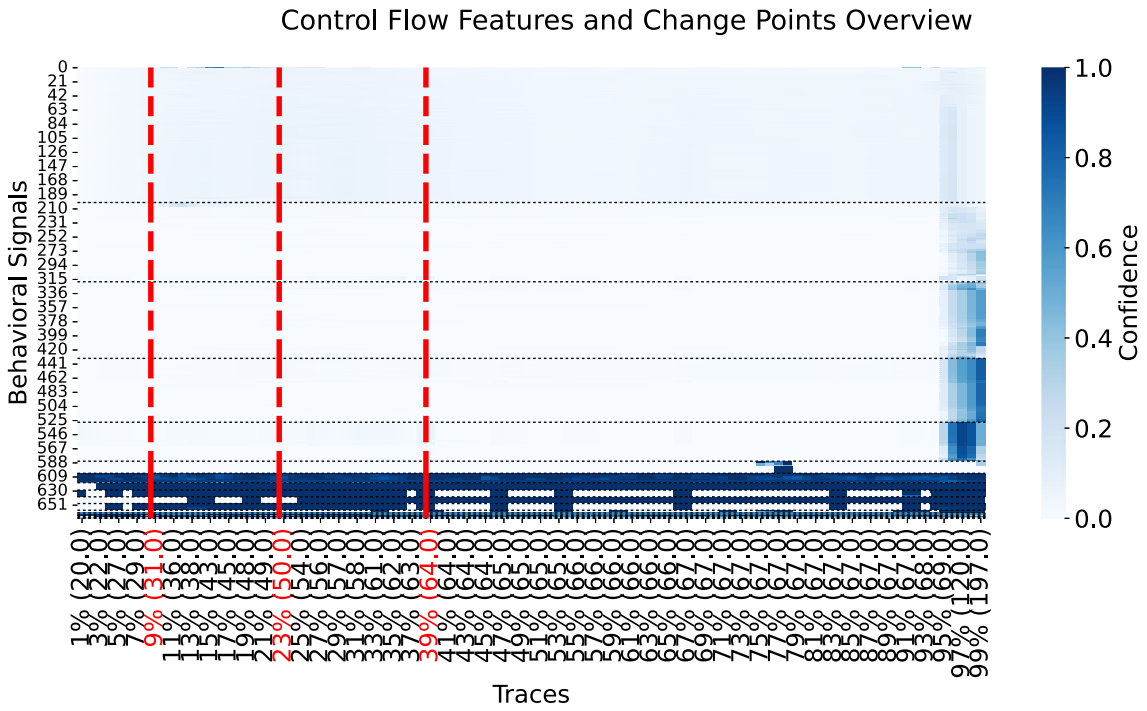


Fig. 21 Visualization of the visual concept drift detection method applied to the accepted cases of the UWV event log

– **Segment 2**: Although the correlation values for this segment are generally low across clusters, visual inspection of the heatmap reveals that cluster 2 begins to emerge from this segment onward. Additionally, the behavioral signals in Clusters 1 and 3 show limited coverage and low confidence in this segment.
– **Segment 3**: This segment exhibits moderately high negative correlation with cluster 6, where corr$(sig_3, clust_6) = -0.61$ and a moderate positive correlations with cluster 4, where corr$(sig_3, clust_4) = 0.66$ and cluster 5, where corr$(sig_3, clust_5)$=0.64.
– **Segment 4**: There is a strong positive correlation between this segment and the behavioral signals in cluster 1, where corr$(sig_4, clust_1) = 0.98$ and cluster 3, where corr$(sig_4, clust_3) = 0.85$, as well as a relatively strong negative correlation with cluster 7, where corr$(sig_4, clust_7) = -0.77$.

Focusing on specific segments and clusters, Table 10 highlights representative behavioral signals that contribute to strong positive or negative correlations. In the following, we provide a summary of key insights derived from the analysis performed using our framework.

Explainability Through Segment-to-Cluster Associations Analyzing the relationship between the identified change points and the clustered behavioral signals, grouped by similar trends, offers valuable insights into the underlying drivers of control-flow changes.

Segment 1 The first segment comprises cases with durations between 0 and 26 days (approximately one month) and shows a strong positive correlation with cluster 7 and a negative correlation with cluster 2. A closer examination reveals that cluster 7 includes constraints such as *"Send Fine must never occur"* and *"If Create Fine occurs, then Payment occurs immediately after"*, indicating streamlined processing. In contrast, cluster 2 includes constraints like *"Create Fine and Send Fine always occur together"* and *"Send Fine occurs at least once"*, reflecting more administrative involvement.

Segment 2 The second segment spans durations from 26 to 145 days (roughly one to five months). In this segment, the behavioral signals in cluster 2 gain high confidence, while those in cluster 7 decline, suggesting that fines are being sent in this period and payments are no longer made immediately. This shift highlights a transition toward less efficient or more delayed processing behavior.

Segment 3 The third segment includes cases with durations between 145 and 340 days (approximately five months to one year). Cluster 3 becomes prominent here and continues to be active in later segments. The correlation analysis indicates a moderate positive relationship with clusters 4 and 5 and a negative correlation with cluster 6. Cluster 3 is characterized by the introduction of the *Add Penalty* event, while clusters 5 and 6 capture opposing behaviors around appeal-related subprocesses. Cluster 5 suggests that appeal activities occur, whereas cluster 6 implies their absence.

Segment 4 The fourth segment comprises cases lasting more than 340 days (over one year). These cases show strong positive correlation with clusters 3 and 4 and a negative correlation with cluster 7. The negative correlation with cluster 7 reflects the breakdown of immediate payment after fine creation. Cluster 3 emphasizes the necessity of penalties, and cluster 1 includes constraints leading to *Send for Credit Collection*, indicating prolonged or incomplete payments that escalate to debt collection procedures.

Comparison to the Baseline Figure 23 visualizes the results obtained using the baseline method. Several of the identified change points show similarities to those detected by our proposed approach. For example, change points around day 20 (versus 26), day 133 (versus 145), and day 424 (versus 340) exhibit considerable alignment. However, some change points identified by the baseline method—specifically those around day 57 and day 869—lack clear or plausible interpretations based on the observed behavioral signals.

5.3.5 BPIC17

Figure 24a presents the results of applying change point detection across the case durations using different window sizes. Across all window sizes, a significant change point is consistently observed at 29 days, indicating that cases with durations longer than one month exhibit substantially different control-flow behavior compared to those with shorter durations. However, this visualization alone does not offer insights into what specific aspects of the process have changed. By selecting a window size of $w = 2$ and a threshold $\theta_{cp} = 0.2$, this change point is effectively identified, and the corresponding behavioral signals are extracted to support interpretability. Figure 24b displays these behavioral signals, with the detected change point marked by a dashed red line. Additionally, Figure 24c illustrates the pairwise comparison of the two resulting segments using the earth mover's distance, which reveals markedly different control-flow patterns.

To enhance the interpretability of the results and highlight which subgroups of behavioral signals contribute to changes in control flow, we applied hierarchical clustering. By selecting 8 clusters, we grouped together behavioral signals that exhibit similar patterns of change. The correlation between these clusters and the identified segments is visualized in Fig. 24d.

- **Segment 1**: This segment exhibits a strong positive correlation with cluster 6, where $\text{corr}(sig_1, clust_6) = 0.85$ and cluster 8, where $\text{corr}(sig_1, clust_8) = 0.81$, along with a moderately high positive correlation with cluster 5, where $\text{corr}(sig_1, clust_5) = 0.71$. It also shows a strong negative correlation with cluster 1, where $\text{corr}(sig_1, clust_1) = -0.88$ and a notable negative correlation with cluster 3, where $\text{corr}(sig_1, clust_3) = -0.71$.
- **Segment 2**: As only two segments are defined, this segment demonstrates complementary correlation patterns to segment 1, with clusters that are positively correlated in Segment 1 showing negative correlations here, and vice versa.

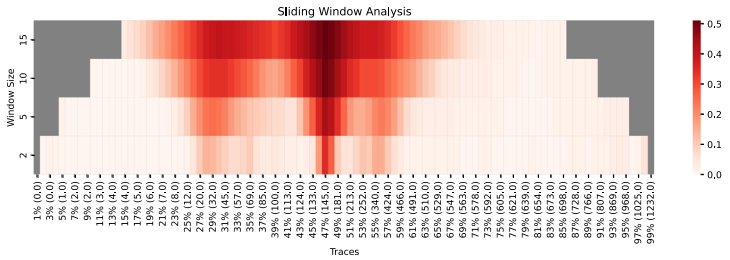
Furthermore, Table 11 reports representative behavioral signals for selected segment-cluster pairs, focusing on those with the highest score values.

Explainability Through Segment-to-Cluster Associations A significant change in control-flow behavior is observed across the duration range of the cases. By combining insights from the trends in behavioral signals with the EMD-based change point detection, we can further investigate what specific characteristics of the control flow have changed and how these changes manifest.

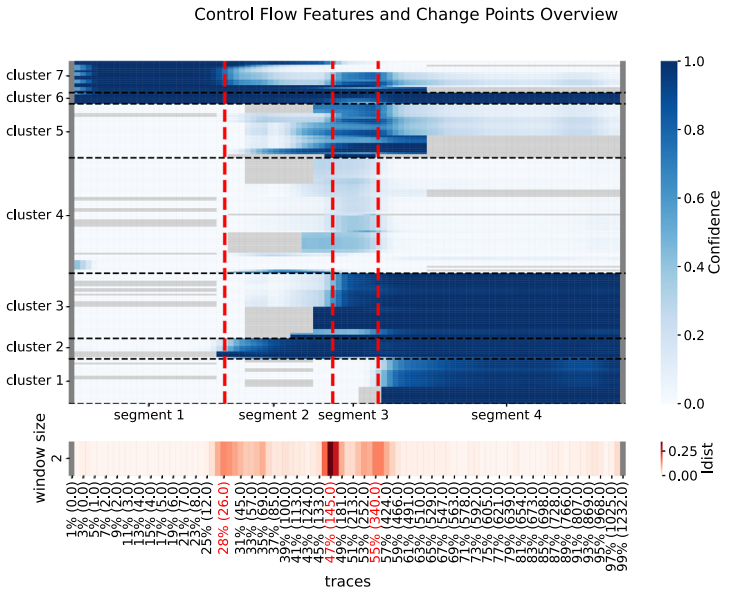
Segment 1 The behavioral signals in cluster 6 predominantly involve relations between *O_Returned* and other activities, suggesting that the subprocess involving this activity plays a key role in cases with durations shorter than 29 days. Similarly, the signals in cluster 8 exhibit a strong positive correlation with cases in the first segment. An examination of the constraints listed in Table 11 reveals that cancellations are relatively rare in this segment, as evidenced by the absence of the activity *A_Cancelled*. In contrast, the frequent occurrence of activities such as *O_Accepted* and *A_Pending* indicates that these cases are generally accepted and are eligible to receive a loan.

Segment 2 The strong correlation between cases in the second segment and clusters 1 and 3 indicates that these cases frequently involve subprocesses related to the cancellation of loan offers and loan applications. As shown in Fig. 24b, clusters 1 and 3 exhibit similar behavioral patterns. The primary distinction is that cluster 3 has higher coverage in the first segment compared to the second, whereas cluster 1 shows low coverage overall, as evidenced by the gray areas in the heatmap. Cluster 3 consists of behavioral signals associated primarily with cancelled loan offers, while cluster 1 includes signals related to the cancellation of both loan offers and loan applications.

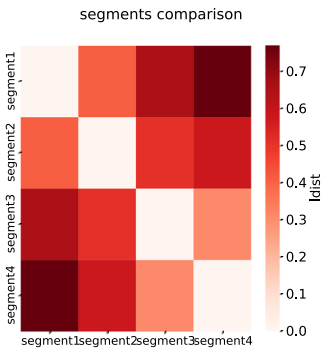
Comparison to the Baseline Figure 25 shows the visualization produced using the baseline method. While a change point is correctly identified at 29 days, consistent with the behavioral signals, two additional change points at 13 and 35 days are also detected, though they do not clearly align with observable patterns in the visualization. In contrast to our EMD-based approach, the baseline method does not offer a straightforward mechanism for assessing the significance of individual change points or determining which ones are more meaningful.



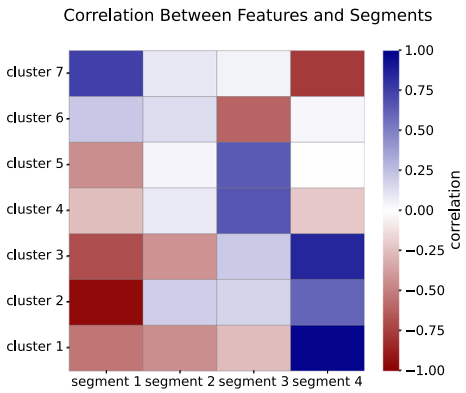
(a) Control flow change detection using the earth mover's distance framework with b=100 and different window sizes $w \in \{2, 5, 10, 15\}$. The color intensity indicates the magnitude of control-flow changes.



(b) Control flow feature clusters derived from behavioral signals and change-point detection. The red dashed lines are the change points detected using the EMD-based framework considering $\theta_{cp} = 0.14$. Gray points correspond to low coverage values ($\theta_{cvg} = 0.05$).



(c) Pairwise comparison of control-flow behaviors across the two identified segments. The earth mover's distance is used to measure the dissimilarities between segments.



(d) Correlation analysis between identified segments and behavioral feature clusters. The heatmap highlights positive and negative correlations, illustrating how specific clusters explain segment-level behaviors.

Fig. 22 Comprehensive analysis of the RTFM event log using the proposed framework

Table 10 Top behavioral signals extracted from specific segment column (seg) and per cluster (sorted by $|score|$, RTFM Event Log)

Seg	Cl	$\lvert S'_L \rvert$	Behavioral signal description	$score$	rank
1	2	7	*Create Fine* and *Send fine* always occur together	−93.0	1
			Send Fine occurs at least once	−92.7	2
1	7	11	*Send Fine* must never occur	92.7	1
			Create Fine occurs, then *Payment* occurs immediately after it	89.0	2
4	1	14	*Send for Credit Collection* occurs only if preceded by *Create fine* with no other send for credit collection in between	97.7	1
			Send for Credit Collection occurs only if preceded by insert fine notification with no other send for credit collection in between	97.7	2
4	3	16	*Add Penalty* occurs at least once	83.6	1
			If *Create Fine* occurs, then *Add penalty* occurs afterward before create Fine recurs	83.6	2
3	5	7	*Insert Date Appeal to Prefecture* and *Send Appeal to Prefecture* occur if and only if they follow one another, alternating	86.2	1
3	6	6	*Payment* occurs at most once	−38.1	1
			Insert Date Appeal to Prefecture must never occur	−17.1	2

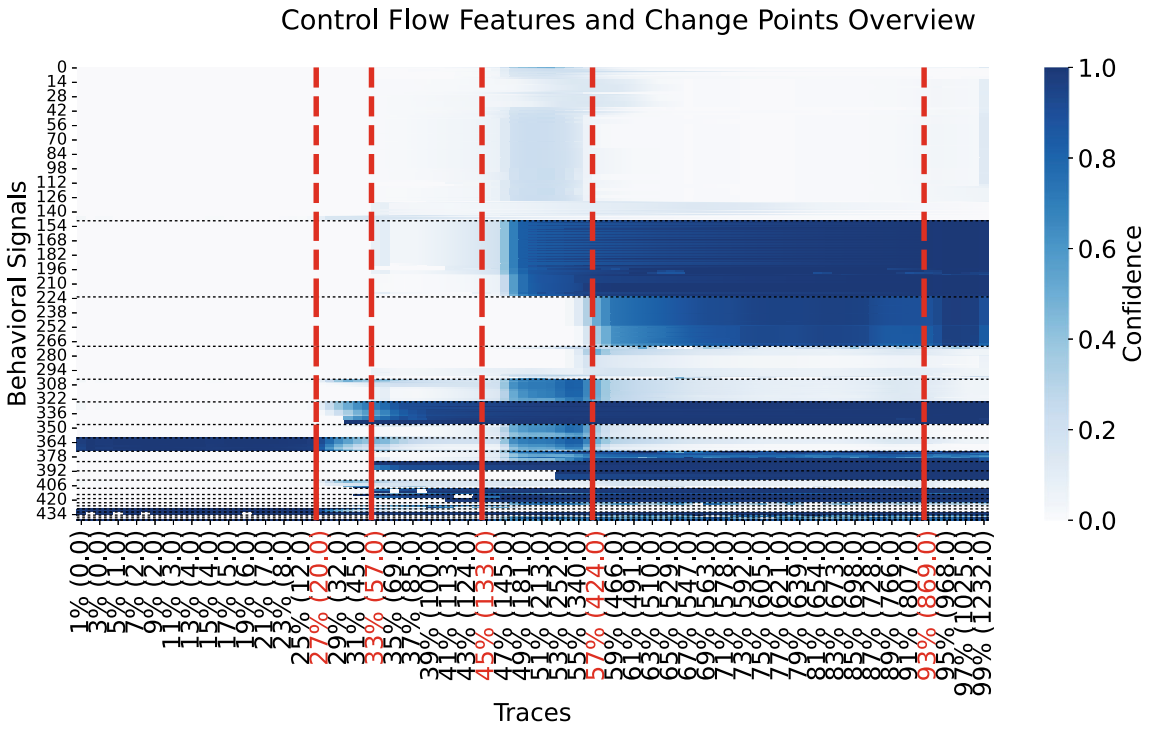


Fig. 23 Visualization of the visual concept drift detection method applied to the RTFM event log

Table 11 Top behavioral signals per cluster (sorted by $|score|$, BPIC17 Event Log)

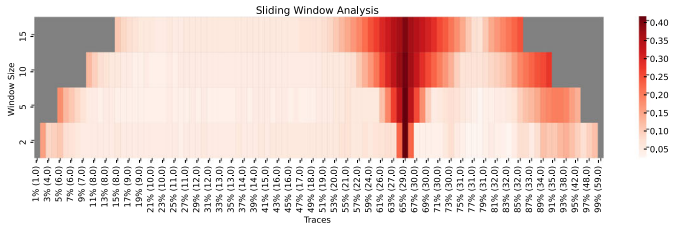
Seg	Cl	$\lvert S'_L \rvert$	Behavioral signal description	*score*	rank
1	6	85	*O_Returned* occurs only if preceded by *A_Validating*	95.3	1
			O_Returned occurs only if preceded by *O_Create offer* with no other *O_Returned* in between	94.6	3
1	8	29	*O_Cancelled* must never occur	57.3	1
			O_Accepted occurs at least once	52.8	2
			A_Pending occurs at least once	52.8	3
2	1	42	*A_Cancelled* and *O_Cancelled* always occur together	82.9	1
			A_Cancelled occurs if and only if it is followed by *O_Cancelled*	81.1	2
			A_Cancelled occurs only if preceded by *O_Create offer* with no other *A_Cancelled* in between	70.6	7
2	3	33	If *O_Sent (mail and online)* occurs, then *O_Cancelled* occurs after it	50.4	1
			If *O_Create offer* occurs, then *O_Cancelled* occurs after it	47.3	3

6 Conclusion

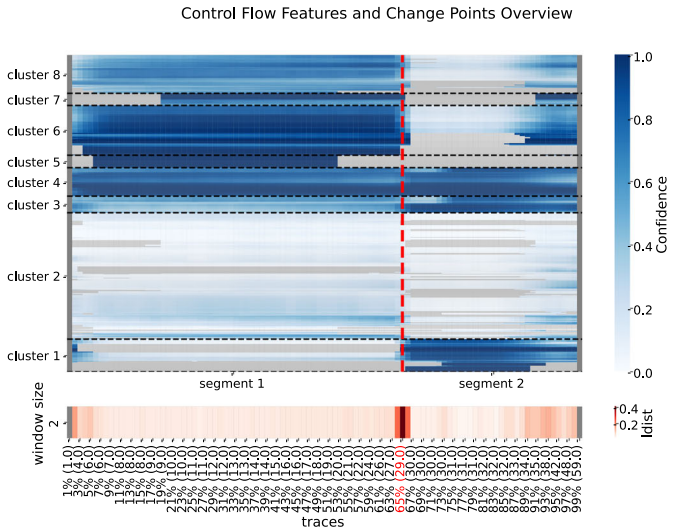
This paper presents a comprehensive framework to enhance explainability in process variant analysis, emphasizing the detection and interpretation of control-flow changes across continuous performance dimensions. By employing the earth mover's distance and a sliding window, the framework accurately identifies change points in control flow, providing a robust foundation for analyzing performance-based process variability. Building on this, the proposed explainability-driven approach allows for a deeper investigation of identified change points. By encoding control-flow behavior into a feature space defined by declarative constraints, the framework correlates variations in feature quality with the identified change points across the continuous dimension, enabling a nuanced understanding of process dynamics.

While the framework effectively balances accuracy and explainability, scalability remains a limitation due to the computational cost associated with the earth mover's distance calculation and the generation and analysis of large feature sets. Future research could focus on optimizing these computationally intensive aspects to enhance efficiency and applicability to larger event logs.

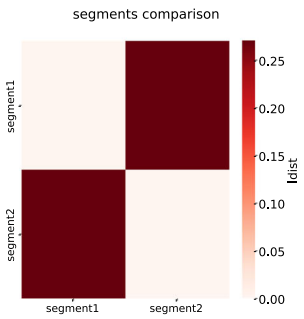
Declarative constraints offer powerful representations of sophisticated relationships between activities. However, exploring alternative methods to encode control flow and



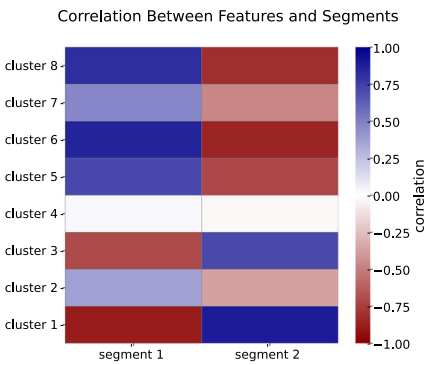
(a) Control flow change detection using the earth mover's distance framework with b=100 and different window sizes $w \in \{2, 5, 10, 15\}$. The color intensity indicates the magnitude of control-flow changes.



(b) Control flow feature clusters derived from behavioral signals and change-point detection. The red dashed lines are the change points detected using the EMD-based framework considering $\theta_{cp} = 0.2$. Gray points correspond to low coverage values ($\theta_{cvg} = 0.05$).



(c) Pairwise comparison of control-flow behaviors across the two identified segments. The earth mover's distance is used to measure the dissimilarities between segments.



(d) Correlation analysis between identified segments and behavioral feature clusters. The heatmap highlights positive and negative correlations, illustrating how specific clusters explain segment-level behaviors.

Fig. 24 Comprehensive analysis of the BPIC17 event log using the proposed framework

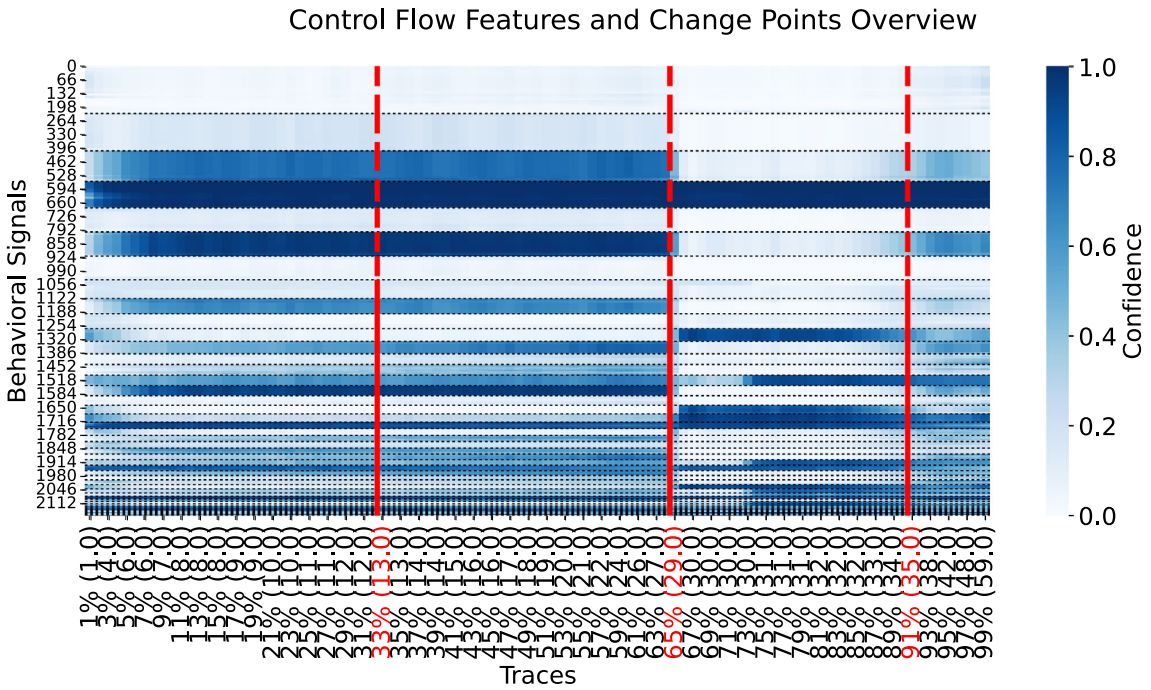


Fig. 25 Visualization of the visual concept drift detection method applied to the BPIC17 event log

integrating them with declarative constraints could yield even more granular and detailed insights into process behavior. Additionally, presenting the identified clusters in an interpretable and concise manner poses challenges. Developing techniques to generate representative samples or summaries for each cluster can significantly improve usability and clarity.

To further assess the practical value of the framework and guide future improvements, conducting user studies with domain experts in real-world settings represents a valuable direction for future work. Such studies could provide empirical insights into the usability, interpretability, and relevance of the extracted explanations. Another promising direction is extending the framework to support multidimensional analysis. Rather than focusing on a single continuous dimension, a multidimensional approach could simultaneously account for multiple performance dimensions, providing a more comprehensive understanding of performance-based process variability and dynamics.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. van der Aalst, W.M.P., Carmona, J.: (eds.): Process Mining Handbook, Lecture Notes in Business Information Processing, vol. 448. Springer (2022)
2. Adams, J.N., van Zelst, S.J., Rose, T., van der Aalst, W.M.P.: Explainable concept drift in process mining. Inf. Syst. **114**, 102177 (2023). https://doi.org/10.1016/J.IS.2023.102177
3. Aghabozorgi, S.R., Shirkhorshidi, A.S., Teh, Y.W.: Time-series clustering - a decade review. Inf. Syst. **53**, 16–38 (2015). https://doi.org/10.1016/J.IS.2015.04.007
4. Back, C.O., Simonsen, J.G.: Comparing trace similarity metrics across logs and evaluation measures. In: Indulska, M., Reinhartz-Berger, I., Cetina, C., Pastor, O. (eds.) Advanced Information Systems Engineering - 35th International Conference, CAiSE 2023, Zaragoza, Spain, June 12-16, 2023, Proceedings. Lecture Notes in Computer Science, vol. 13901, pp. 226–242. Springer (2023). https://doi.org/10.1007/978-3-031-34560-9_14
5. Bergami, G., Francescomarino, C.D., Ghidini, C., Maggi, F.M., Puura, J.: Exploring business process deviance with sequential and declarative patterns. CoRR **abs/2111.12454** (2021)
6. Bolt, A., van der Aalst, W.M.P.: Multidimensional process mining using process cubes. In: Gaaloul, K., Schmidt, R., Nurcan, S., Guerreiro, S., Ma, Q. (eds.) Enterprise, Business-Process and Information Systems Modeling - 16th International Conference, BPMDS 2015, 20th International Conference, EMMSAD 2015, Held at CAiSE 2015, Stockholm, Sweden, June 8-9, 2015, Proceedings. Lecture Notes in Business Information Processing, vol. 214,

pp. 102–116. Springer (2015). https://doi.org/10.1007/978-3-319-19237-6_7
7. Bolt, A., de Leoni, M., van der Aalst, W.M.P.: Process variant comparison: using event logs to detect differences in behavior and business rules. Inf. Syst. **74**(Part), 53–66 (2018). https://doi.org/10.1016/J.IS.2017.12.006
8. Brockhoff, T., Uysal, M.S., van der Aalst, W.M.P.: Time-aware concept drift detection using the earth mover's distance. In: van Dongen, B.F., Montali, M., Wynn, M.T. (eds.) 2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4-9, 2020. pp. 33–40. IEEE (2020). https://doi.org/10.1109/ICPM49681.2020.00016
9. Buliga, A., Vazifehdoostirani, M., Genga, L., Lu, X., Dijkman, R.M., Francescomarino, C.D., Ghidini, C., Reijers, H.A.: Uncovering patterns for local explanations in outcome-based predictive process monitoring. In: Marrella, A., Resinas, M., Jans, M., Rosemann, M. (eds.) Business Process Management - 22nd International Conference, BPM 2024, Krakow, Poland, September 1-6, 2024, Proceedings. Lecture Notes in Computer Science, vol. 14940, pp. 363–380. Springer (2024). https://doi.org/10.1007/978-3-031-70396-6_21
10. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018)
11. Ceravolo, P., Comuzzi, M., De Weerdt, J., Di Francescomarino, C., Maggi, F.M.: Predictive process monitoring: concepts, challenges, and future research directions. Process Sci. **1**(1), 1–22 (2024)
12. Chesani, F., Francescomarino, C.D., Ghidini, C., Grundler, G., Loreti, D., Maggi, F.M., Mello, P., Montali, M., Tessaris, S.: Shape your process: Discovering declarative business processes from positive and negative traces taking into account user preferences. In: Almeida, J.P.A., Karastoyanova, D., Guizzardi, G., Montali, M., Maggi, F.M., Fonseca, C.M. (eds.) Enterprise Design, Operations, and Computing - 26th International Conference, EDOC 2022, Bozen-Bolzano, Italy, October 3-7, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13585, pp. 217–234. Springer (2022). https://doi.org/10.1007/978-3-031-17604-3_13
13. Ciccio, C.D., Maggi, F.M., Montali, M., Mendling, J.: Resolving inconsistencies and redundancies in declarative process models. Inf. Syst. **64**, 425–446 (2017). https://doi.org/10.1016/J.IS.2016.09.005
14. Ciccio, C.D., Mecella, M.: On the discovery of declarative control flows for artful processes. ACM Trans. Manag. Inf. Syst. **5**(4), 24:1-24:37 (2015). https://doi.org/10.1145/2629447
15. Ciccio, C.D., Montali, M.: Declarative process specifications: Reasoning, discovery, monitoring. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook, Lecture Notes in Business Information Processing, vol. 448, pp. 108–152. Springer (2022). https://doi.org/10.1007/978-3-031-08848-3_4
16. Francescomarino, C.D., Donadello, I., Ghidini, C., Maggi, F.M., Rizzi, W., Tessaris, S.: Making sense of temporal event data:a framework for comparing techniques for the discovery of discriminative temporal patterns. In: Guizzardi, G., Santoro, F.M., Mouratidis, H., Soffer, P. (eds.) Advanced Information Systems Engineering - 36th International Conference, CAiSE 2024, Limassol, Cyprus, June 3-7, 2024, Proceedings. Lecture Notes in Computer Science, vol. 14663, pp. 423–439. Springer (2024). https://doi.org/10.1007/978-3-031-61057-8_25
17. Hompes, B., Buijs, J., van der Aalst, W.M.P., Dixit, P., Buurman, J.: Discovering deviating cases and process variants using trace clustering. In: Proceedings of the 27th Benelux Conference on Artificial Intelligence (BNAIC). pp. 5–6 (2015)
18. Leemans, S.J.J., McGree, J.M., Polyvyanyy, A., ter Hofstede, A.H.M.: Statistical tests and association measures for business processes. IEEE Trans. Knowl. Data Eng. **35**(7), 7497–7511 (2023). https://doi.org/10.1109/TKDE.2022.3197840
19. Leemans, S.J.J., Shabaninejad, S., Goel, K., Khosravi, H., Sadiq, S.W., Wynn, M.T.: Identifying cohorts: Recommending drill-downs based on differences in behaviour for process mining. In: Dobbie, G., Frank, U., Kappel, G., Liddle, S.W., Mayr, H.C. (eds.) Conceptual Modeling - 39th International Conference, ER 2020, Vienna, Austria, November 3-6, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12400, pp. 92–102. Springer (2020). https://doi.org/10.1007/978-3-030-62522-1_7
20. Leemans, S.J.J., Syring, A.F., van der Aalst, W.M.P.: Earth movers' stochastic conformance checking. In: Hildebrandt, T.T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) Business Process Management Forum - BPM Forum 2019, Vienna, Austria, September 1-6, 2019, Proceedings. Lecture Notes in Business Information Processing, vol. 360, pp. 127–143. Springer (2019). https://doi.org/10.1007/978-3-030-26643-1_8
21. de Leoni, M., van der Aalst, W.M.P.: Data-aware process mining: discovering decisions in processes using alignments. In: SAC. pp. 1454–1461. ACM (2013)
22. de Leoni, M., van der Aalst, W.M.P., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Inf. Syst. **56**, 235–257 (2016). https://doi.org/10.1016/J.IS.2015.07.003
23. Maaradji, A., Dumas, M., Rosa, M.L., Ostovar, A.: Detecting sudden and gradual drifts in business processes from execution traces. IEEE Trans. Knowl. Data Eng. **29**(10), 2140–2154 (2017). https://doi.org/10.1109/TKDE.2017.2720601
24. Müllner, D.: Modern hierarchical, agglomerative clustering algorithms. CoRR arXiv:abs/1109.2378 (2011)
25. Norouzifar, A., van der Aalst, W.M.P.: Discovering process models that support desired behavior and avoid undesired behavior. In: Hong, J., Lanperne, M., Park, J.W., Cerný, T., Shahriar, H. (eds.) Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, SAC 2023, Tallinn, Estonia, March 27-31, 2023. pp. 365–368. ACM (2023). https://doi.org/10.1145/3555776.3577818
26. Norouzifar, A., Dees, M., van der Aalst, W.M.P.: Imposing rules in process discovery: An inductive mining approach. In: Araújo, J., de la Vara, J.L., Santos, M.Y., Assar, S. (eds.) Research Challenges in Information Science - 18th International Conference, RCIS 2024, Guimarães, Portugal, May 14-17, 2024, Proceedings, Part I. Lecture Notes in Business Information Processing, vol. 513, pp. 220–236. Springer (2024). https://doi.org/10.1007/978-3-031-59465-6_14
27. Norouzifar, A., Rafiei, M., Dees, M., van der Aalst, W.M.P.: Process variant analysis across continuous features: A novel framework. In: van der Aa, H., Bork, D., Schmidt, R., Sturm, A. (eds.) Enterprise, Business-Process and Information Systems Modeling - 25th International Conference, BPMDS 2024, and 29th International Conference, EMMSAD 2024, Limassol, Cyprus, June 3-4, 2024, Proceedings. Lecture Notes in Business Information Processing, vol. 511, pp. 129–142. Springer (2024). https://doi.org/10.1007/978-3-031-61007-3_11
28. Richetti, P.H.P., Jazbik, L.S., Baião, F., Campos, M.L.M.: Deviance mining with treatment learning and declare-based encoding of event logs. Expert Syst. Appl. **187**, 115962 (2022). https://doi.org/10.1016/J.ESWA.2021.115962
29. Sato, D.M.V., Freitas, S.C.D., Barddal, J.P., Scalabrin, E.E.: A survey on concept drift in process mining. ACM Comput. Surv. **54**(9), 189:1-189:38 (2022). https://doi.org/10.1145/3472752
30. Slaats, T., Debois, S., Back, C.O.: Weighing the pros and cons: Process discovery with negative examples. In: Polyvyanyy, A., Wynn, M.T., Looy, A.V., Reichert, M. (eds.) Business Process Management - 19th International Conference, BPM 2021, Rome, Italy, September 06-10, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12875, pp. 47–64. Springer (2021). https://doi.org/10.1007/978-3-030-85469-0_6

31. Taymouri, F., Rosa, M.L., Dumas, M., Maggi, F.M.: Business process variant analysis: survey and classification. Knowl. Based Syst. **211**, 106557 (2021). https://doi.org/10.1016/J.KNOSYS.2020.106557
32. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: review and benchmark. ACM Trans. Knowl. Discov. Data **13**(2), 17:1-17:57 (2019). https://doi.org/10.1145/3301300
33. Weerdt, J.D., vanden Broucke, S.K.L.M., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12), 2708–2720 (2013). https://doi.org/10.1109/TKDE.2013.64
34. Yeshchenko, A., Ciccio, C.D., Mendling, J., Polyvyanyy, A.: Visual drift detection for event sequence data of business processes. IEEE Trans. Vis. Comput. Graph. **28**(8), 3050–3068 (2022). https://doi.org/10.1109/TVCG.2021.3050071
35. Yeshchenko, A., Mendling, J.: A survey of approaches for event sequence analysis and visualization. Inf. Syst. **120**, 102283 (2024). https://doi.org/10.1016/J.IS.2023.102283
36. Zandkarimi, F., Rehse, J., Soudmand, P., Hoehle, H.: A generic framework for trace clustering in process mining. In: van Dongen, B.F., Montali, M., Wynn, M.T. (eds.) 2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4-9, 2020. pp. 177–184. IEEE (2020). https://doi.org/10.1109/ICPM49681.2020.00034
37. van Zelst, S.J., Cao, Y.: A generic framework for attribute-driven hierarchical trace clustering. In: del-Río-Ortega, A., Leopold, H., Santoro, F.M. (eds.) Business Process Management Workshops - BPM 2020 International Workshops, Seville, Spain, September 13-18, 2020, Revised Selected Papers. Lecture Notes in Business Information Processing, vol. 397, pp. 308–320. Springer (2020). https://doi.org/10.1007/978-3-030-66498-5_23

Ali Norouzifar is a PhD student at RWTH Aachen University in the Process and Data Science (PADS) group. He holds both a Master's and a Bachelor's degree in Control Engineering from Isfahan University of Technology. His PhD research is mainly part of the DataNinja Research Training Group, funded by the German federal state of North Rhine-Westphalia. His main research interests include event data analysis and process discovery. In his work, he investigates how the control flow of processes can vary across different perspectives and leverages this knowledge to discover process models that more accurately represent a given perspective.

Marcus Dees has worked at Uitvoeringsinstantie Werknemersverzekeringen (UWV) as Customer Intelligence Analist. UWV is the Employee Insurance Agency of the Netherlands. Since 2025 Marcus works as an integral capacity management consultant at Radboud university medical center (Radboudumc) in Nijmegen, the Netherlands. In his work, Marcus connects present-day business problems with scientific research in the Business Process Management and the Process Mining domain.

Majid Rafiei is a Senior Process and Data Scientist and received his PhD in Process and Data Science from the Chair of Process and Data Science (PADS) at RWTH Aachen University. He also holds a Master of Engineering in Electronic Commerce from Amirkabir University of Technology, Tehran. His research interests include Process Mining, Machine Learning, Responsible Data Science, and Information Retrieval.

Wil van der Aalst is an Alexander von Humboldt professor at RWTH Aachen University, leading the Process and Data Science (PADS) group. He is also the Chief Scientist at Celonis, part-time affiliated with the Fraunhofer FIT. His research interests include process mining, Petri nets, business process management, workflow automation, and data science. According to Research.com, he is the second-highest-ranked computer scientist in Germany and ranked 9th worldwide (2025 ranking). Van der Aalst is an IFIP Fellow, IEEE Fellow, and ACM Fellow, as well as an elected member of the Royal Netherlands Academy of Arts and Sciences, the Royal Holland Society of Sciences and Humanities, the Academy of Europe, and the German Academy of Science and Engineering.