Defining and visualizing process execution variants from partially ordered event data

Daniel Schuster [a,b,*], Francesca Zerbato [c], Sebastiaan J. van Zelst [a,b], Wil M.P. van der Aalst [a,b]

[a] *Fraunhofer Institute for Applied Information Technology FIT, Data Science and Artificial Intelligence, Schloss Birlinghoven, Sankt Augustin, 53757, Germany*

[b] *RWTH Aachen University, Chair of Process and Data Science, Aachen, Germany*

[c] *University of St. Gallen, Institute of Computer Science, St. Gallen, Switzerland*

A B S T R A C T

The execution of operational processes generates event data stored in enterprise information systems. Process mining techniques analyze such event data to obtain insights vital for decision-makers to improve the reviewed process. In this context, event data visualizations are essential. We focus on visualizing variants describing process executions that are control flow equivalent. Such variants are an integral concept for process mining and are used, for instance, for data exploration and filtering. We propose high-level and low-level variants covering different levels of abstraction and present corresponding visualizations. Compared to existing variant visualizations, we support partially ordered event data and allow for heterogeneous temporal information per event, i.e., we support both time intervals and time points. We evaluate our contributions using automated experiments showing practical applicability to real-life event data. Finally, we present a user study indicating significantly improved usefulness and ease of use of the proposed high-level variant visualization compared to existing variant visualizations for typical analysis tasks.

1. Introduction

Event data are generated during the execution of processes—ranging from administrative to production processes—and stored in enterprise information systems. *Process mining* [1] comprises techniques for analyzing event data to gain insights into the underlying process. These insights, such as automatically discovered process models [2,3], conformance diagnostics [4–6], or temporal performance diagnostics [7–9], support decision-making regarding process modifications and ultimately support process improvements.

Since event data include many process executions, each containing several process activities (hereinafter *activities*), *execution variants* (hereinafter *variants*) are a key abstraction for handling high volumes of event data. Variants describe process executions whose activities share identical order relations. Thus, a one-to-many relationship between variants and process executions exists.

Variant analysis allows process analysts to understand the differences between process executions based on their control flow or performance [10]. Further, analysts can filter event data based on variants satisfying specific conditions [11]. During exploratory analysis, *variant visualizations* help to understand the occurrence and order of frequent activity patterns and the level of process

* Corresponding author.
E-mail address: daniel.schuster@fit.fraunhofer.de (D. Schuster).

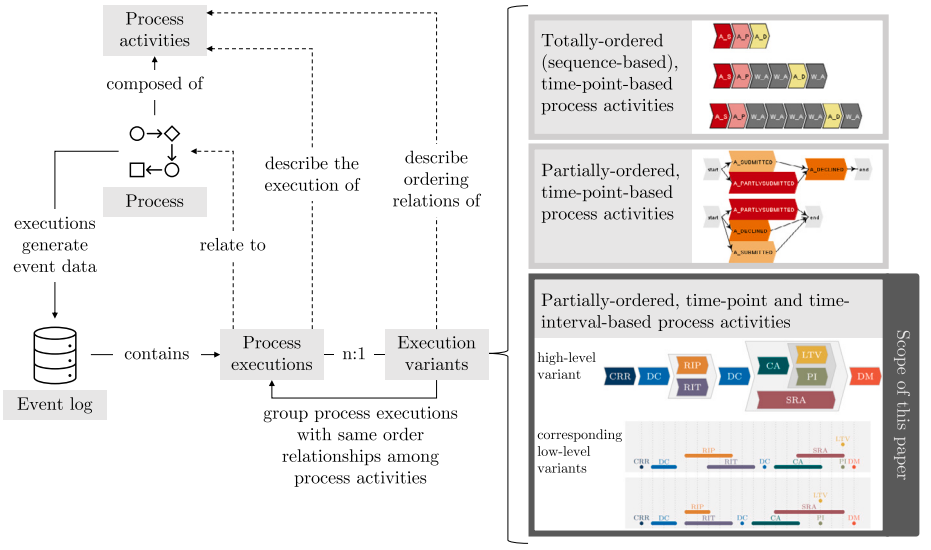


Fig. 1. Overview of key process mining terms, including variants that group process executions with the same ordering relationships with respect to the activities. Three different approaches to consider variants, including the contribution of this paper.

structuredness [12]. Such insights help analysts to assess the standardization potential of the process [13] and make informed decisions about the kinds of analyses to be performed on a specific event log [14]. In short, variant visualizations are essential for successfully applying process mining.

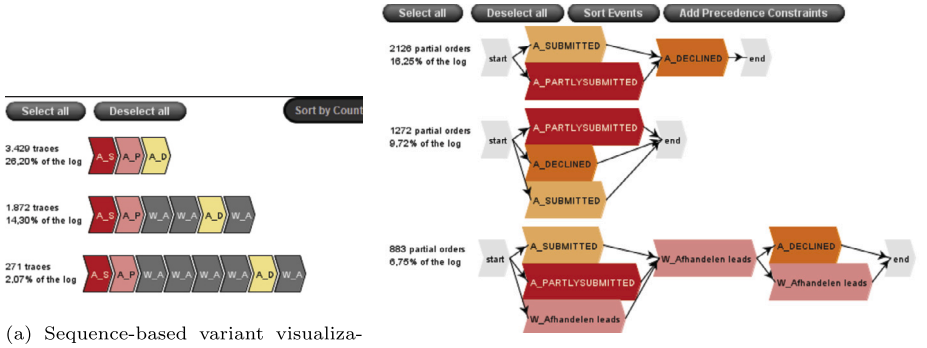
Variants are often defined for *totally-ordered* single-timestamped activities [1]. Fig. 1 shows in the top right a widely used variant visualization consisting of totally-ordered, time-point-based activities represented as a sequence of colored chevrons. In this work, we use the term *parallel* in the context of the execution of activities when their execution somehow overlaps in time. Parallel activities are a common phenomenon in processes. However, variants based on totally ordered activities lack the expressiveness to capture them properly. For instance, different activities within a process execution having identical timestamps must be sequentialized.[1] Such forced sequentialization, however, can lead to incorrect conclusions by analysts, who are often unaware of the sequentialization made by software tools. In [15], variants consisting of partially ordered, time-point-based activities were proposed; Fig. 1 shows the corresponding chevron-based visualization of two example variants. These variants can capture parallel activities; however, they consider activities time-point-based, i.e., atomic. Consequently, these variants lack expressiveness regarding the execution span of activities.

This paper considers partially ordered event data with *heterogeneous temporal information* about the executed activities. Thus, activities are either atomic, representing a time point [16], or double-timestamped, i.e., start and completion timestamps exist, representing a time interval [17]. We focus on both point-based and interval-based activity executions since the level of detail in recording activities can vary; heterogeneous temporal information is a regular phenomenon in event data capturing actual processes [18–20]. We propose two definitions and visualizations for variants that comprise partially ordered, time-point, and interval-based activities. Both definitions address different abstraction levels and thus complement each other. We also investigate the impact of adjustments in temporal granularity, often necessary to obtain the right abstraction level for a given process mining project, on the proposed variants. More specifically, we address the following research questions (RQs).

RQ1 How to define variants for event data that are partially ordered and can contain heterogeneous temporal information per activity?
RQ2 How to visualize process variants for partially ordered event data with heterogeneous temporal information?
RQ3 How do temporal granularity changes of the event data affect the defined variants?

The contributions of this paper are as follows. Regarding RQ1 and RQ2, we build upon prior work [21] in which we introduced a preliminary variant definition for time-interval-based activities. In this paper, we generalize the variant definition proposed in [21] and introduce a revised definition that considers time-point and time-interval-based activities, i.e., we focus on event data *with heterogeneous temporal information*. We refer to this variant definition and visualization as *high-level variants*, cf. Fig. 1. Moreover, we propose a novel variant definition and visualization referred to as *low-level variants*, cf. Fig. 1. Low-level variants are a complementary variant definition addressing a lower abstraction level than high-level variants, allowing one to see the temporal ordering of activities in more detail. Regarding RQ3, we discuss the impact of temporal granularity modifiers on variants. We quantitatively evaluate the

[1] Activities' timestamps are usually used as a first-order criterion. If this is insufficient, a second-order criterion is needed (often simply the order of events in the data is used).



(a) Sequence-based variant visualization using the visualization *Explore Event Log*.

(b) Partial-order-based variant visualization using the visualization *Explore Partial Orders* [15].

Fig. 2. Screenshots of ProM [35] showing variant visualizations of a real-life event log [18].

proposed variant definitions using real-life event logs via automated experiments. Moreover, we present a user study that assesses the perceived usefulness and usability of the proposed high-level variant visualization compared to existing visualizations. This study also evaluates how well analysts interpret event data using the proposed visualization compared to existing ones.

The subsequent sections are organized as follows. Sect. 2 outlines related work, while Sect. 3 introduces preliminary concepts. Sect. 4 addresses RQ1 and RQ2 by introducing definitions and visualizations for variants. Sect. 5 discusses the effect of temporal granularity changes on variants, i.e., RQ3. Sect. 6 presents an implementation of our contributions. Sect. 7 describes the evaluation of the proposed variants, including automated experiments and the user study. Sect. 8 concludes this paper.

2. Related work

This section starts with a brief overview of related work on process mining visualizations, focusing on event data visualizations in Sect. 2.1. Subsequently, in Sect. 2.2, we cover related work about variant definitions and their visualizations.

2.1. Event data visualizations in process mining

Data visualizations are of great importance for any data analysis [22]. Likewise, process mining analysis strongly relies on the exploration of visual artifacts, e.g., process models that are visualized as BPMN models [23], Petri nets [24] or directly-follows graphs [25]. In addition to process models, visualizations are used to illustrate the temporal executions of processes, e.g., borrowing from the area of time-oriented data visualization [26]. However, the time complexity of the event data used for process mining makes the visualization of their temporal structure and order an open challenge [27].

Still, various visualizations and visual analytics [28] techniques for event data have been proposed within process mining. Although a complete overview is outside this paper's scope, we report examples of commonly used techniques below.

A well-known visualization is the *dotted chart* [29], which visualizes events from an event log as dots in a 2-dimensional axis system and allows analysts to configure the dimensions from the available event attributes freely. Dotted charts may reveal diverse patterns, such as the temporal development of case arrivals and the distribution of case cycle times. The *performance spectrum* [30] visualizes the time differences between the execution of activities. Thus, analysts can discover bottlenecks and batch processing of activities in a process. In [31], the authors propose an approach to visualize clusters of traces to facilitate analysts working with trace clustering techniques. Compared to the notion of variants considered in this work, trace clustering techniques, e.g., [32,33], consider various other trace features irrelevant for variants as contemplated in this paper, for instance, resource information of activities. In [34], the authors propose a visual approach for concept drift detection in event data. The visual approach allows analysts to detect and analyze process drifts from event logs containing sequence-based traces. For visualizing individual process executions, time charts are commonly used, as shown in Fig. 3c, cf. [23, Fig. 11.16] and [1, Fig. 9.12]. The examples show that a broad spectrum of visualization techniques is used in process mining. This paper, however, focuses on *variants*, i.e., unique process executions regarding the ordering of activities. In the following section, we focus on variant definitions and corresponding visualizations.

2.2. Variant definitions and corresponding visualizations

As described in Sect. 1 and illustrated in Fig. 1, process executions and variants are often defined as a sequence of activities [1]. As a result, many process mining tools feature sequence-based variant visualizations. For example, Fig. 2a shows a common visualization of sequence-based variants, i.e., unique sequences of activity labels, in the process mining tool ProM [35]. Since many process mining techniques assume traces containing totally ordered activities, there are even methods that convert non-sequential process executions, i.e., partially ordered event data as considered in this work, into the most likely sequential execution [36]. Event sequence visualization has also been addressed by diverse works outside of process mining; we refer to [37] for an extensive overview of

Table 1
Example of an event log representing the execution of a mortgage application process.

ID			Timestamp		
Activity instance	Case	Activity label	Start	Completion	...
678253	1	credit request received (CRR)	⊥	16.06.21 12:43:35	...
678254	1	document check (DC)	17.06.21 08:32:23	18.06.21 12:01:11	...
678247	1	request information from applicant (RIP)	19.06.21 09:34:00	22.06.21 09:12:00	...
678248	1	request information from third parties (RIT)	19.06.21 14:54:00	25.06.21 08:57:12	...
678249	1	document check (DC)	⊥	28.06.21 14:23:59	...
678250	1	credit assessment (CA)	30.06.21 13:02:11	04.07.21 08:11:32	...
678251	1	security risk assessment (SRA)	01.07.21 17:23:11	06.07.21 18:51:43	...
678252	1	property inspection (PI)	⊥	05.07.21 00:00:00	...
678253	1	loan-to-value ratio determined (LTV)	⊥	05.07.21 00:00:00	...
678254	1	decision made (DM)	⊥	08.07.21 14:13:18	...
678255	2	credit request received (CRR)	⊥	17.06.21 23:21:31	...
...	...	...	...	...	...

approaches and characterization of event sequence visualization techniques. However, work on defining and visualizing variants that contain non-sequential process executions, i.e., partially ordered event data, as considered in this work, is limited.

In [15], the authors consider variants consisting of partially-ordered, time-point-based activities. Fig. 2b shows the corresponding variant visualization. Compared to our approach, each activity is considered atomic, i.e., it is described by a single timestamp.[2] Thus, when visualizing variants comprising double-timestamped activities, the visualization splits such activity instances into two single-timestamped activities, i.e., the start and the completion.

In prior work [21], as mentioned in Sect. 1, we propose a variant definition for partially-ordered time-interval-based activities. In this paper, we extend this definition and define high-level variants that consider heterogeneous temporal information, i.e., partially-ordered time-point and time-interval-based activities. Furthermore, we propose a novel second variant definition, i.e., low-level variants, that indicates more details regarding the temporal order of activities within a variant than high-level variants. Note that there is a one-to-many relation between high-level and low-level variants, i.e., one high-level variant might subsume many low-level variants. We propose a corresponding visualization for both variant definitions and present the impact of time granularity changes.

This paper considers partially ordered event data with heterogeneous temporal information, i.e., some activities represent time intervals, and others represent time points, cf. Fig. 1. In this context, there is general work on relationships between time intervals. Allen's interval algebra [38] defines thirteen possible relations in which time intervals can be related. The critical difference between this algebra and the two variant definitions proposed in this paper is that we consider both time intervals and time points. Allen's interval algebra consciously focuses on relations between proper time intervals, i.e., intervals where the start point is strictly smaller than the end time point. Also, Allen's algebra does not provide visualizations for combined relationships between multiple intervals.

3. Preliminaries

This section introduces concepts and definitions used throughout this paper.

3.1. Event data

Event data describe the execution of processes. Event data from the same process is referred to as an *event log*; Table 1 shows an example of a mortgage application process. Each row describes an *event*, i.e., an *activity instance*. For example, the first event indicates that for case 1 activity 'credit request received' has been executed on 16.06.21 12:43:35. Note that the event does not contain a start timestamp, denoted by the symbol ⊥.[3] The second event indicates that activity 'document check' was started on 17.06.21 at 08:32:23 and completed on 18.06.21 at 12:01:11. Note that the event log contains heterogeneous temporal information; activity instances represent either time points or intervals.

Henceforth, we denote the universe of activity instance identifiers by $\mathcal{I}^A$, case identifiers by $\mathcal{I}^C$, timestamps by positive real numbers $\mathbb{R}^+$, and the universe of activity labels by $\mathcal{L}$. Next, we define activity instances.

Definition 1 *(Activity instances).* An activity instance $a = (i, c, l, t_s, t_c) \in \mathcal{I}^A \times \mathcal{I}^C \times \mathcal{L} \times \left(\mathbb{R}^+ \cup \{\bot\}\right) \times \mathbb{R}^+$, uniquely identified by $i \in \mathcal{I}^A$, represents the execution of an activity $l \in \mathcal{L}$ that was executed for the process instance identified by $c \in \mathcal{I}^C$. The activity instance's temporal information is given by the start timestamp $t_s \in \mathbb{R}^+ \cup \{\bot\}$ that may be missing, and the complete timestamp $t_c \in \mathbb{R}^+$. Further, $t_s \neq \bot \Rightarrow t_s \leq t_c$. We denote the universe of activity instances by $\mathcal{A}$.

To refer to specific components of an activity instance $a = (i, c, l, t_s, t_c) \in \mathcal{A}$, we use short forms: a^i, a^c, a^l, a^{t_s}, and a^{t_c}. Next, we define an event log as a set of activity instances that describe the *same* process.

[2] The chevrons' lengths vary due to the activity labels and do not indicate execution times.
[3] For simplicity, we always use the completion timestamp for time-point-based activities.

Definition 2 *(Event log).* An event log $E \subseteq \mathcal{A}$ consists of a finite set of activity instances, each having a unique identifier, describing the same process.

The term *trace* refers to individual process executions. Thus, a trace is the set of all activity instances belonging to the same *case*.

Definition 3 *(Trace).* Let $E \subseteq \mathcal{A}$ be an event log. A trace $T \subseteq E$ describing case $c \in \mathcal{I}^C$ is defined as $T = \{a \in E \mid a^c = c\}$. We denote the traces of E as $\mathcal{T}^E$.

3.2. Partial orders

We use strict partial orders to relate activity instances within a trace to define variants eventually.

Definition 4 *(Strict partial order).* Let P be a set and $\prec \subseteq P \times P$ be a binary relation. We call $(P, \prec)$ a strict partial order iff for arbitrary elements $a, b, c \in P$ it holds that: $a \not\prec a$ (Irreflexivity), $a \prec b \Rightarrow b \not\prec a$ (Asymmetry), and $(a \prec b \wedge b \prec c) \Rightarrow a \prec c$ (Transitivity). We denote the universe of strict partial orders by $\mathcal{P}$.

Finally, we define isomorphism for traces whose activity instances are partially ordered, cf. Definition 4. Two traces are isomorphic if their activity instances, reduced to the activity label, have the same order relations.

Definition 5 *(Trace order isomorphism).* Let E be an event log and $(T_1, \prec_1), (T_2, \prec_2) \in \mathcal{P}$ with $T_1, T_2 \in \mathcal{T}^E$ being traces. $(T_1, \prec_1)$ is isomorphic to $(T_2, \prec_2)$, denoted by $(T_1, \prec_1) \cong (T_2, \prec_2)$, iff a bijective function $f : T_1 \to T_2$ exists such that: $\forall a \in T_1 \left(a^l = f(a)^l\right)$ and $\forall a, a' \in T_1 \left(a \prec_1 a' \Leftrightarrow f(a) \prec_2 f(a')\right)$.

4. Defining and visualizing variants

This section introduces two variant definitions, i.e., high-level and low-level variants, and the corresponding visualizations. First, we provide an overview of these variants using a running example in Sect. 4.1. Then, we formally introduce high-level variants in Sect. 4.2, and low-level variants in Sect. 4.3. Finally, we discuss the limitations of the proposed variant visualizations in Sect. 4.4.

4.1. Overview

We propose two variant definitions, i.e., high-level and low-level, for event data with heterogeneous temporal information. A *one-to-many* relation exists between high-level and low-level variants, as the behavior described by one high-level variant may comprise multiple low-level ones. This two-level variant hierarchy facilitates the exploration of temporal relations at different levels of detail.

Fig. 3 exemplifies the relation between traces, low-level, and high-level variants. The traces shown in Fig. 3c contain both single-timestamped and double-timestamped activity instances, i.e., heterogeneous temporal information. Colors are used to distinguish different activity labels. Activities can occur multiple times within a trace and might have different temporal information per execution. For example, activity DC occurs in all four traces twice; however, in each trace, once single-timestamped and once double-timestamped (cf. Fig. 3c).

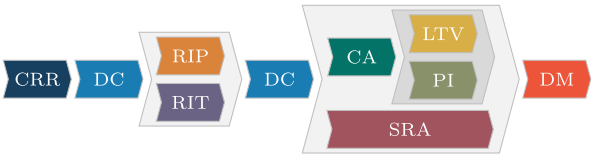
Fig. 3a depicts the high-level variant that comprises the four shown traces. High-level variants show the ordering relations between activities. Whenever the execution of activities somehow overlaps or happens simultaneously, chevrons representing the activities are placed on top of each other, indicating parallelism. The visualized variant indicates that the traces start with activity CRR, followed by DC. Next, activities RIP and RIT are executed parallel, followed by DC. Next, while SRA is executed, CA is executed, followed by a parallel execution of LTV and PI. Eventually, DM is executed. Note that high-level variants do not visually distinguish between single- and double-timestamped activity instances. Variants generally abstract temporal information since the x-axis indicates neither absolute nor relative time. Thus, variant visualizations do not display information about activities' execution time, i.e., only order relations are shown.

Fig. 3b shows the corresponding low-level variants that provide a more detailed view of the four traces than high-level variants. We observe that two low-level variants describe the four traces compared to a single high-level variant. Low-level variant 1 describes traces T_1 and T_3. Low-level variants distinguish between time-point-based activities and interval-based activities. We observe that activity CRR, for which there is only a single timestamp in both traces, is executed first, followed by DC, which has a double timestamp and is therefore drawn as an interval. Further, low-level variants visualize the exact relationships between activities in more detail. For instance, in the first variant, RIP and RIT are started simultaneously; however, RIP ends before RIT.

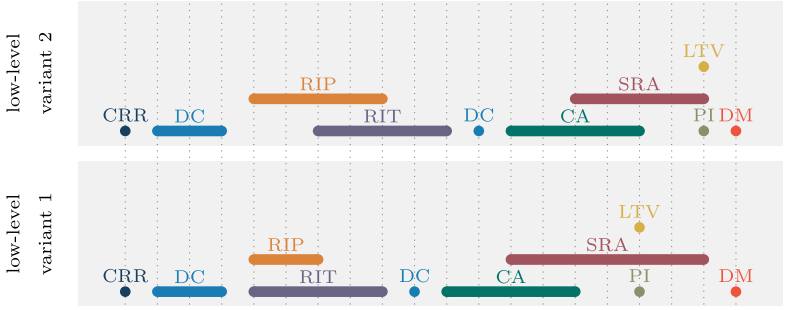
In the two subsequent sections, we define high-level (cf. Sect. 4.2) and low-level variants (cf. Sect. 4.3). Both sections are structured as follows. First, we introduce the variant definition. Subsequently, we present a corresponding visualization.

4.2. High-level variants

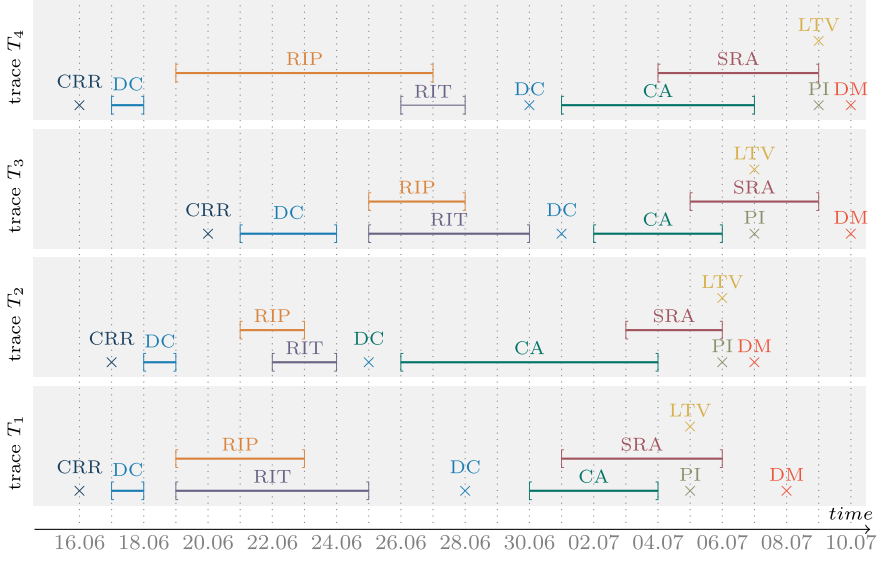
This section defines high-level variants and their visualization, cf. Fig. 3a. In short, a high-level variant focuses on parallel and sequential orderings of activities. When the execution of activities overlaps—the exact classification of the overlap is irrelevant



(a) Variant level: High-level variant whose described process behavior comprises the behavior described by the two low-level variants (Fig. 3b) and the four traces (Fig. 3c). High-level variants are introduced in Sect. 4.2.



(b) Variant level: Low-level variants whose described behavior comprises the traces in Fig. 3c. Low-level variant 1 includes traces T_1 and T_3; low-level variant 2 includes traces T_2 and T_4. Low-level variants are introduced in Sect. 4.3.



(c) Trace level: Visualization of four process executions, i.e., traces. Each trace consists of activities with one or two timestamps each, represented by points (×) or intervals ([–]), respectively.

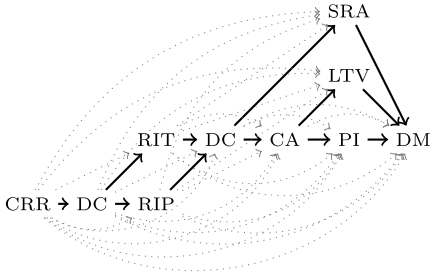
Fig. 3. Process executions and corresponding high- and low-level variants from a mortgage application process; activity labels are abbreviated according to Table 1.

compared to low-level variants—they are considered parallel. Otherwise, activities are considered sequentially ordered. Fig. 3a shows a high-level variant comprising the four traces shown in Fig. 3c.

Definition 6 *(High-level trace view).* Let $E \subseteq \mathcal{A}$ be an event log and $T = \{a_1, \ldots, a_n\} \in \mathcal{T}^E$ be a trace. The high-level trace view of T is a strict partial order $(T, \prec_{HL}) \in \mathcal{P}$. For arbitrary $a_1, a_2 \in T$ it holds that $a_1 \prec_{HL} a_2$ iff:

- $a_1^{t_c} < a_2^{t_s}$ if $a_1^{t_c}, a_2^{t_s} \in \mathbb{R}^+$, or
- $a_1^{t_c} < a_2^{t_c}$ if $a_1^{t_c} \in \mathbb{R}^+$ and $a_2^{t_s} = \perp$.

(a) Trace T_1 as shown in Fig. 3c.



(b) Graph representation of $(T_1, \prec_{HL})$.

Fig. 4. Example of a trace and its corresponding high-level trace view.

For example, Fig. 4 visualizes the high-level trace view of T_1, i.e., $(T_1, \prec_{HL})$; vertices represent the activity instances and are labeled with the corresponding activity labels. Next, we generally define a variant for arbitrary partial orders.

Definition 7 *(Variant).* Let $E \subseteq \mathcal{A}$ be an event log with traces $\mathcal{T}^E = \{T_1, \dots, T_n\}$. Let $(T_1, \prec_1), \dots, (T_n, \prec_n)$ be strict partial orders. A variant $V \subseteq \mathcal{T}^E$ describes traces that are isomorphic, i.e., $\forall T_i, T_j \in V \left((T_i, \prec_i) \cong (T_j, \prec_j)\right)$.

For example, when considering the high-level trace view from Definition 6, traces $T_1, \dots, T_4$ (cf. Fig. 3c) are all isomorphic, i.e., $(T_1, \prec_{HL}) \cong \dots \cong (T_4, \prec_{HL})$. Consider Fig. 4b showing the graph representation of $(T_1, \prec_{HL})$; the graph representation of $(T_2, \prec_{HL}), \dots, (T_4, \prec_{HL})$ would look equal. Thus, they all belong to the *same high-level variant*.

4.2.1. Visualizing high-level variants

This section presents a layout algorithm for high-level variants. The input is a partial order according to Definition 6, and the output is a visualization indicating the ordering relations between activities, cf. Fig. 3a. We have chosen chevrons to visualize activities since chevrons are widely used among process mining tools for this purpose (cf. Sect. 2); other geometric shapes are, of course, possible. Below, we introduce sequential and parallel partitions necessary for computing the variant visualization, cf. Fig. 3a. A sequential partition splits up elements in a given partial order $(X, \prec)$ into disjoint subsets $X_1 \cup \dots \cup X_n = X$ such that any two elements from different subsets are related to each other.

Definition 8 *(Sequential partition).* Let $(X, \prec) \in \mathcal{P}$ be a partial order and $X_1, \dots, X_m$ be a partition of X. We call $(X_1, \prec_1), \dots, (X_m, \prec_m)$ a sequential partition of $(X, \prec)$ if it satisfies the conditions below.

- $\forall 1 \leq i < j \leq m \ \forall x_i \in X_i \ \forall x_j \in X_j \ \left(x_i \prec x_j\right)$
- $\forall 1 \leq i \leq m \ \forall x, x' \in X_i \ \left(x \prec_i x' \Leftrightarrow x \prec x'\right)$

Analogously, a parallel partition splits up elements from a given partial order $(X, \prec)$ into disjoint subsets $X_1 \cup \dots \cup X_m = X$ such that any two elements from different subsets are not related to each other.

Definition 9 *(Parallel partition).* Let $(X, \prec) \in \mathcal{P}$ be a partial order and $X_1, \dots, X_m$ be a partition of X. We call $(X_1, \prec_1), \dots, (X_m, \prec_m)$ a parallel partition of $(X, \prec)$ if it satisfies the conditions below.

- $\forall 1 \leq i < j \leq m \ \forall x_i \in X_i \ \forall x_j \in X_j \ \left(x_i \not\prec x_j \wedge x_j \not\prec x_i\right)$
- $\forall 1 \leq i \leq m \ \forall x, x' \in X_i \ \left(x \prec_i x' \Leftrightarrow x \prec x'\right)$

Given a parallel/sequential partition $(X_1, \prec_1), \dots, (X_m, \prec_m)$ of $(X, \prec)$, we call $(X_1, \prec_1), \dots, (X_m, \prec_m)$ a *maximal* parallel/sequential partition if there exists no $k > m$ such that $(X'_1, \prec'_1), \dots, (X'_k, \prec'_k)$ is a parallel/sequential partition of $(X, \prec)$. We refer to [21] for formal proofs that maximal parallel/sequential partitions are unique for partial orders.

The essence of the layout algorithm is to partition a high-level trace view, i.e., a partial order, recursively until no further partitioning can be found, resulting in a hierarchical structure. Each partitioning corresponds to a set of chevrons and their positioning.

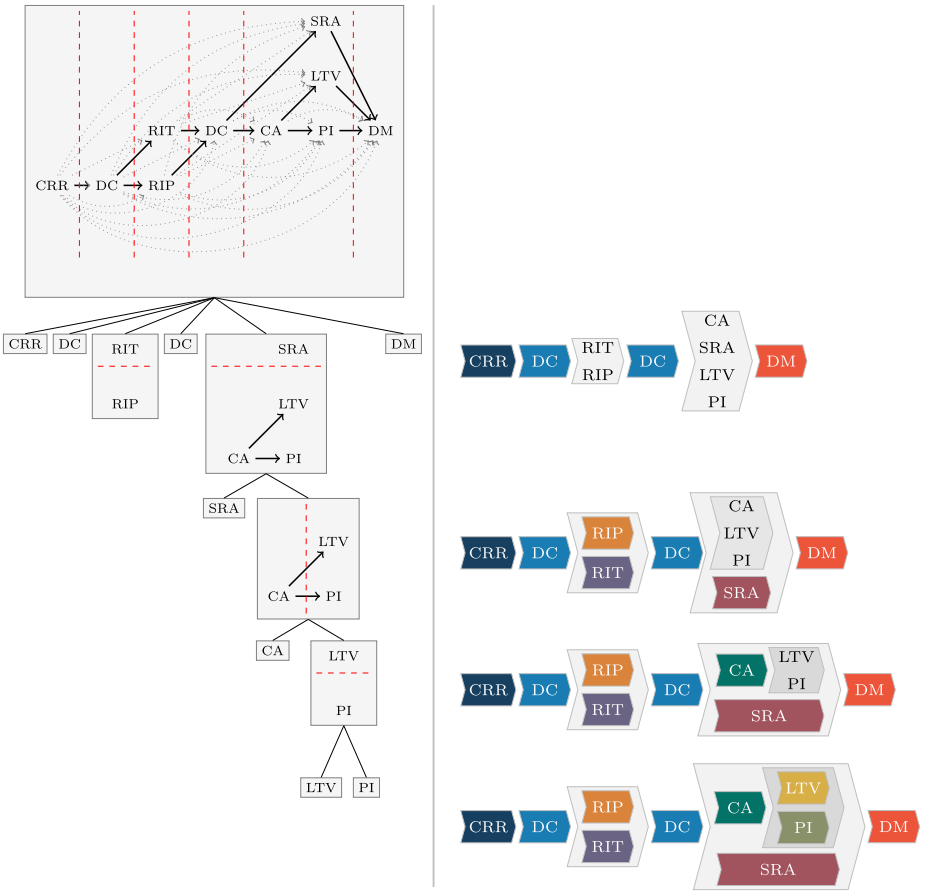


Fig. 5. Recursively partitioning of $(T_1, \prec_{HL})$, shown on the left side. Horizontal red lines symbolize parallel partitions, and vertical red lines symbolize sequential partitions. After each recursion level, the right side shows the corresponding (intermediate) variant visualization.

A sequential partition of size m results in m horizontally aligned chevrons. Likewise, a parallel partition of size m results in m vertically aligned chevrons. Note that, at most, one maximal partitioning is applicable in each step.[4] The left side of Fig. 5 shows the recursive partitioning of the partial order, cf. Fig. 4, and the right side shows the layout calculation. Red dashed lines indicate the recursive partitioning. When a chevron contains a single activity, we color the chevron to help distinguish activities. The final visualization is depicted in the bottom right of Fig. 5.

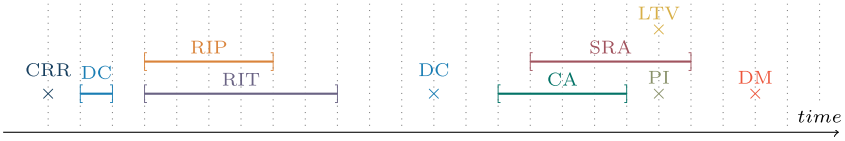
4.3. Low-level variants

This section presents an alternative to high-level variants. While high-level variants solely focus on whether activities somehow overlap, low-level variants classify the overlaps, e.g., is one interval entirely contained in another interval. Next, we define the low-level trace view as a partial order of a trace's activity instances, each split into two elements: the *start* and the *completion*.

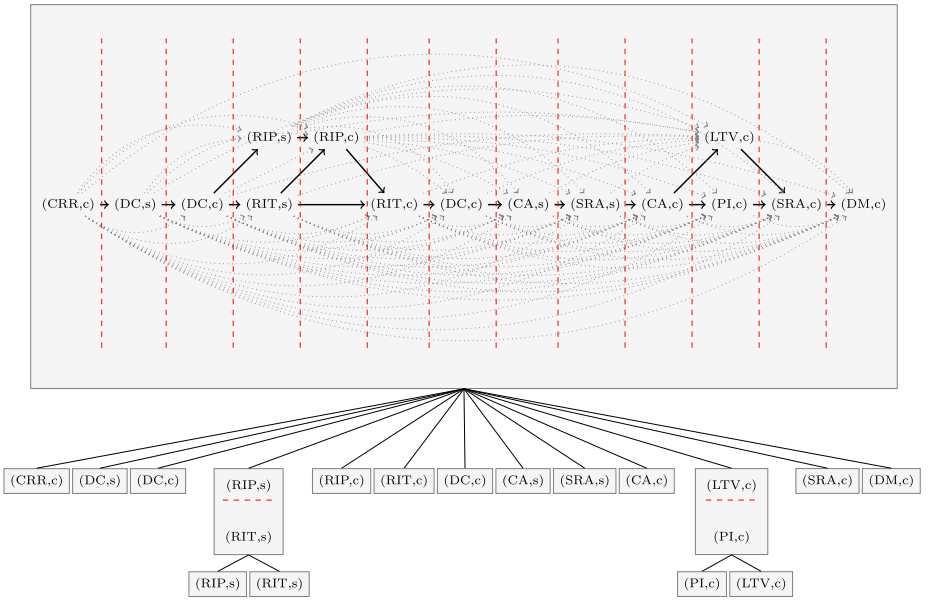
Definition 10 *(Low-level trace view).* Let $E \subseteq \mathcal{A}$ be an event log and let $T \in \mathcal{T}^E$ be a trace. We define $\bar{T} \subseteq T \times \{s, c\}$ as $\bar{T} = \{(a, s) \mid a \in T \wedge a^{t_s} \neq \perp\} \cup \{(a, c) \mid a \in T\}$. The low-level trace view of T is a strict partial order $(\bar{T}, \prec_{LL}) \in \mathcal{P}$. For arbitrary $(a_1, x_1), (a_2, x_2) \in \bar{T}$ it holds that $(a_1, x_1) \prec_{LL} (a_2, x_2)$ iff $a_1^{t_{x_1}} < a_2^{t_{x_2}}$.

Fig. 6b shows at the top the low-level trace view of T_1. According to Definition 10, each activity instance having two timestamps is split into two elements. The node labels in Fig. 6b indicate the corresponding activity label as well as if the node represents the start (s) or the completion (c) of the respective activity. Using the strict partial order definition from Definition 10, the low-level trace views of T_1 and T_3 are isomorphic; thus, T_1 and T_3 are represented by the same low-level variant (cf. Fig. 3).

[4] The proposition has been formally proven in [21]. Note that [21] denotes a sequential partition as an ordering cut and a parallel partition as a parallel cut.

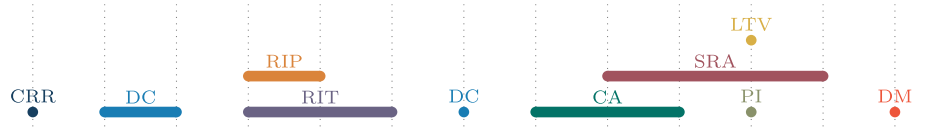


(a) Trace T_1 as shown in Fig. 3c.

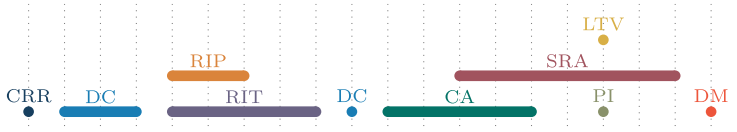


(b) Recursive partitioning of $(T_1, \prec_{LL})$. Horizontal red lines symbolize parallel partitions, and vertical red lines symbolize sequential partitions.

(c) Visualization step 1) Placing start and end nodes according to the calculated partitioning, cf. Fig. 6b



(d) Visualization step 2) Connecting start and end nodes of activity instances

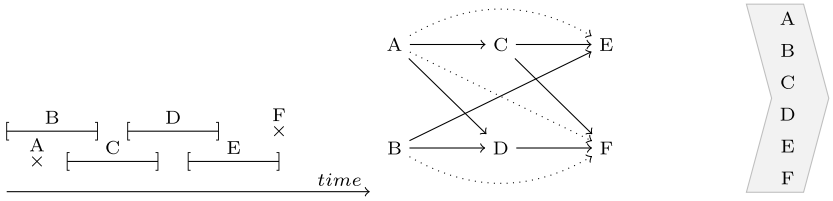


(e) Visualization step 3) Reducing the horizontal distance between intervals or times between which nothing intervenes.

Fig. 6. Example of the construction of a low-level variant visualization.

4.3.1. Visualizing low-level variants

This section proposes a visualization for low-level variants. Analogously to high-level variants (cf. Sect. 4.2.1), we recursively partition the partial order representing the low-level variant. Compared to high-level variants, the recursion depth is at most two: Parallel partitioning or sequential partitioning followed by parallel partitioning. To visually distinguish high-level from low-level variants, we use points and vertical bars instead of chevrons.



(a) Trace showing a pattern of chained activity instances. (b) Corresponding high-level trace view. (c) Corresponding visualization.

Fig. 7. High-level variant where no partition can be found.

Fig. 6 exemplifies the layout calculation of the low-level variant describing trace T_1. Fig. 6b shows the recursive partitioning of $(T_1, \prec_{LL})$ by identifying sequential and parallel partitions. Similar to high-level variants, recursive partitioning determines the placement of elements. Initially, we use dots for each element, cf. Fig. 6d. Sequential partitioning results in horizontally aligned dots and parallel partitions result in vertically aligned dots. Next, we connect dots that belong to the same activity instance, cf. Fig. 6d. Finally, the variant visualization is post-processed by reducing the horizontal distance between the visual elements if no other elements are placed in this area, cf. Fig. 6e.

4.4. Discussion

This section discusses the abstraction level of the high-level variant visualization. In the example shown in Fig. 5, partitions are recursively applied until singletons remain. However, partitions cannot always be found in specific scenarios, even though sets contain more than one element. In these scenarios, the proposed visualization cannot show the exact relationship among all activities contained in the variant. Fig. 7a shows an example trace, and Fig. 7b shows the corresponding high-level trace view. No partitions, neither sequential nor parallel partitions, can be found. Thus, the visualization for this variant corresponds to a gray-colored chevron listing the activity names. The gray chevron indicates that the contained activities occur in an unspecified order. Note that such patterns, where no partitions can be found, can occur at any recursion level.

Low-level variants are not affected by the above-described phenomenon because all low-level trace view elements solely represent time points. Thus, patterns, as exemplified in Fig. 7, cannot occur; for instance, two elements are related to each other (e.g., B and D in Fig. 7) and a third element (e.g., C in Fig. 7) is unrelated to both of them. Low-level variants can thus be seen as an extension that is especially useful when high-level variants cannot accurately represent activity patterns due to their level of abstraction.

5. Adjusting time granularity

An appropriate level of time abstraction is essential to gain valuable insights from event data. Most information systems use discrete time domains with (milli)seconds as the smallest unit [26], e.g., the UNIX time format. This smallest time unit is denoted as the *bottom granularity* [26]. Depending on the objectives, analyzing event data at the millisecond level may be impractical and lead to incorrect conclusions. Reconsider the event log from a mortgage application process, cf. Table 1. Since the process executions have cycle times of several days, the analysis at the level of milliseconds is too fine-grained. For instance, it may be irrelevant or even misleading whether the activity “Request information from applicant” was started a few minutes before or after “Request information from third party”. Instead, the information that both activities were executed on the same day, i.e., parallel from a control flow perspective, is sufficient. Thus, analysts are interested in variants indicating such parallelism.

In some instances, a low bottom granularity adds no value to the process analysis; on the contrary, it may even lead to wrong conclusions. Moreover, the recorded timestamps may not accurately reflect reality. Reconsider the mortgage application process example. Suppose an employee decides on several applications over time and enters all decisions at once into an information system. Suppose the information system cannot track the start of the decision process. Consequently, the completion timestamps for the activity “decision made” (DM) do not accurately reflect reality; the timestamps merely reflect the entry of the activity’s result into the information system. Further, no duration information of the “decision made” is available because no start timestamp is recorded.

The mentioned examples illustrate the need to adjust the temporal granularity when applying process mining. We define the adjustment of the time granularity as a function modifying the timestamps in an event log.

Definition 11 *(Time granularity modifier (TGM)).* $f^{TGM} : \mathbb{R}^+ \to \mathbb{R}^+$ is called a TGM if for any $t_1, t_2 \in \mathbb{R}^+ : t_1 < t_2 \Rightarrow f^{TGM}(t_1) \leq f^{TGM}(t_2)$.

Fig. 8 shows different time granularity modifiers applied to trace T_1, containing activity instances with case identifier 1 (cf. Table 1). The first time granularity modifier (cf. hours in Fig. 8) cuts off each timestamp’s minutes and seconds. The second one maps timestamps onto days. Thereby, e.g., the two start timestamps of the activity RIT and RIP become identical. In comparison, RIP

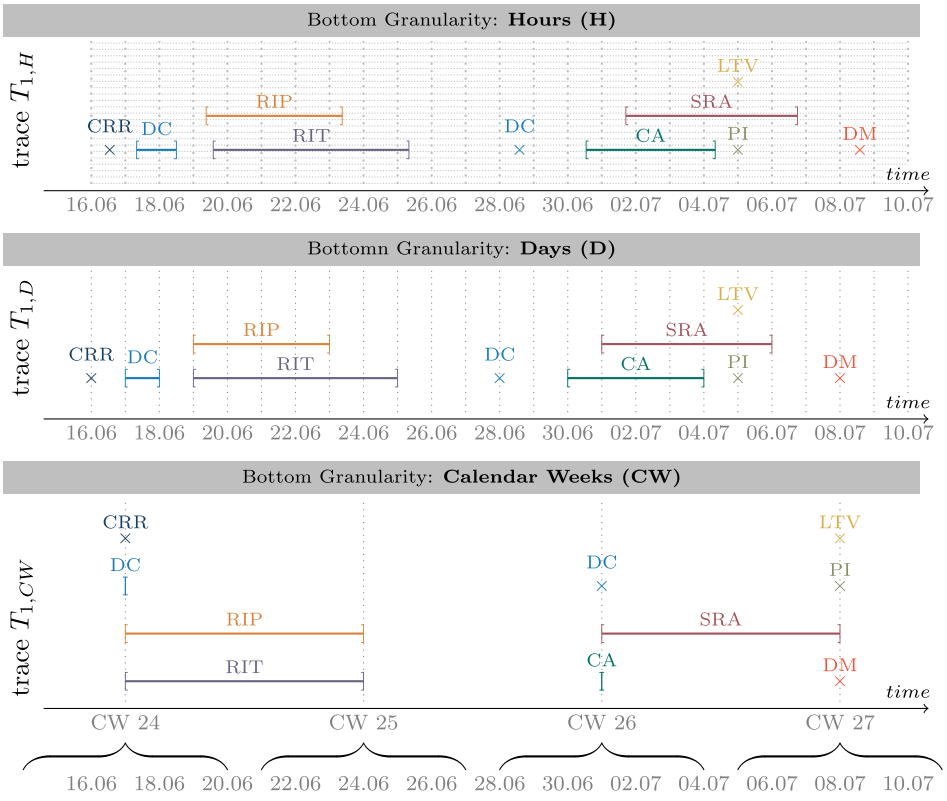


Fig. 8. Different time granularity modifiers applied to trace T_1 consisting of activity instances with case identifier 1, cf. event log shown in Table 1.

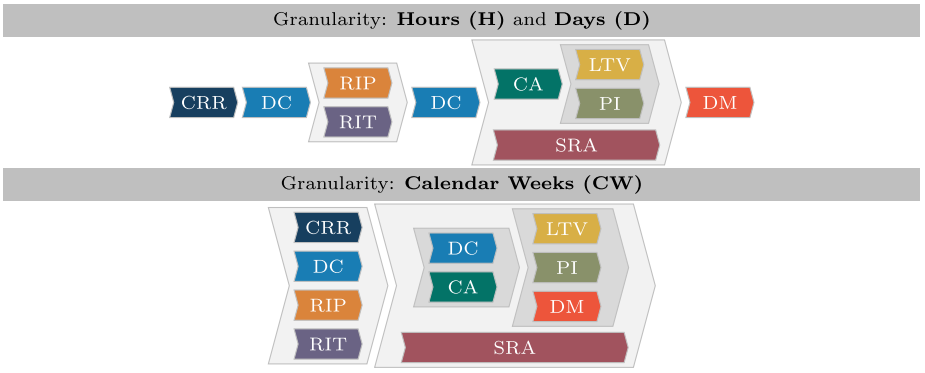


Fig. 9. High-level variants for T_1 per time granularity modifier, cf. Fig. 8.

starts before RIT when looking at the original data (cf. Table 1) or when applying setting the bottom granularity to hours. In short, time granularity modifiers change the timestamps of the different activity instances of a trace and hence might change the order relations among them.

Depending on the chosen time granularity, different variants are obtained. We exemplify the effect of different time granularity modifiers on the variant describing trace T_1. Consider Fig. 9. Both traces $T_{1,H}$ and $T_{1,D}$ are represented by the same high-level variant; however, switching to calendar weeks, i.e., $T_{1,CW}$, affects the corresponding high-level variant, cf. Fig. 9.

Regarding low-level variants, we observe that in the example, each time granularity modifier leads to a different variant, cf. Fig. 10. For example, the slightly earlier start of activity RIP compared to activity RIT in $T_{1,H}$ (cf. Fig. 8) disappears when using days as the time granularity modifier, cf. $T_{1,D}$ in Fig. 8. Switching from a fine to a coarser temporal granularity generally results in more parallel activities. Visually speaking, variants grow vertically and shrink horizontally with coarser time granularity.

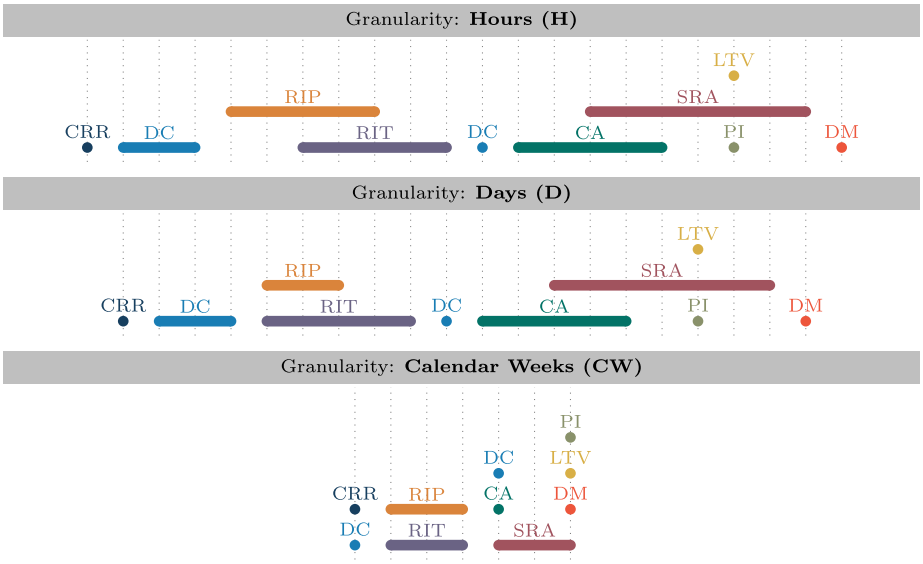
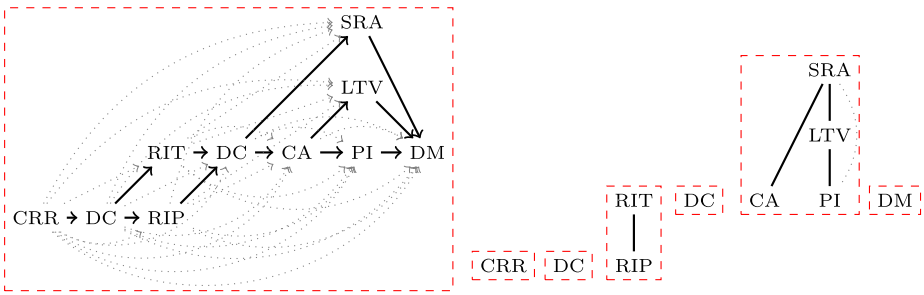


Fig. 10. Low-level variants for T_1 per time granularity modifier, cf. Fig. 8.



(a) High-level trace view graph (identical to the previous example, cf. Fig. 4). The graph shown contains one component, which is highlighted by a red dashed box.

(b) Unrelated graph w.r.t. the high-level trace view graph shown on the left (cf. Fig. 12a). The graph shown contains six components, which are highlighted by red dashed boxes.

Fig. 11. Example of a high-level trace view graph and corresponding unrelated graph.

6. Tool support

This section presents an implementation of the proposed variant visualizations in the open-source process mining tool Cortado [39]. First, we briefly outline the tool generally. Sect. 6.1 introduces the implementation of the variant computation, i.e., the recursive portioning of high-level and low-level trace views and the grouping of traces to variants. Finally, Sect. 6.2 presents the implementation of the proposed variant visualizations in Cortado's user interface.

Cortado is a tool for incremental process discovery [40]. Unlike automatic process discovery [2], user involvement during the discovery of a process model is essential. Thereby, variant visualizations are critical as users gradually select variants to be added to a process model under construction. We refer to [39] for a detailed introduction to Cortado. In the following, we present the implementation of the proposed variants.

6.1. Computing variants

This section covers the implementation of the variant computation, including: the recursive partitioning as presented in Sect. 4 and the merging of traces to variants. We reduce the problem of computing maximal sequential and parallel partitions (cf. Definition 8 and Definition 9) to the well-known problem of computing components of a graph [41]. As introduced in Sect. 4, we first create the high-level or low-level trace view (depending on which variant is being computed) and save this view as a graph. In the remainder

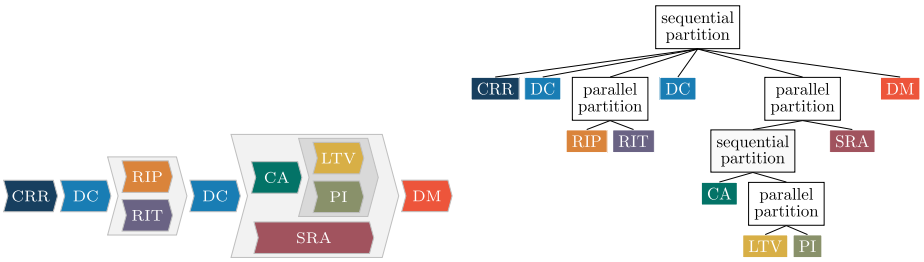
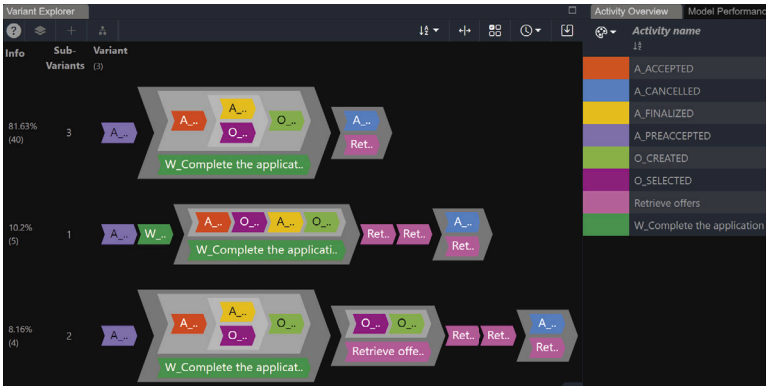
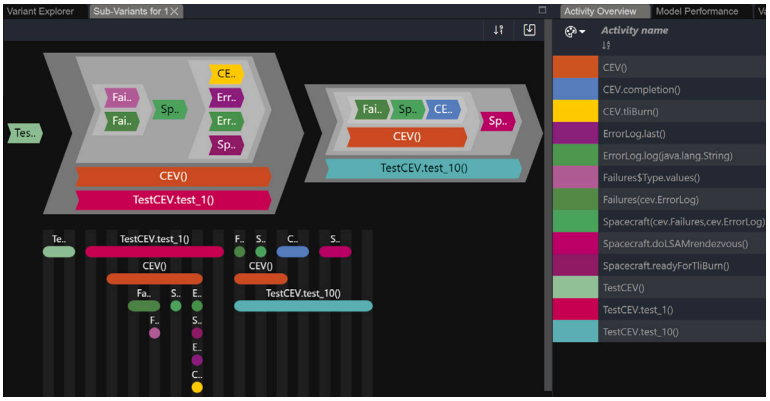


Fig. 12. Tree representation of a high-level variant (cf. Fig. 3a).



(a) High-level variants from an event log.



(b) A high-level variant and a corresponding low-level variant from an event log.

Fig. 13. Screenshots of Cortado's variant explorer visualizing event data.

of this section, we refer to the graph representing the high-level/low-level trace view simply as the trace view graph. Additionally, we store the corresponding, so-called *unrelated graph* to each trace view graph. The unrelated graph contains the same nodes as the corresponding trace view graph but only connects any two nodes that are not related in the corresponding trace view. For instance, Fig. 11a depicts a high-level trace view graph, and Fig. 11b depicts the corresponding unrelated graph. Subsequently, we look for components in the trace view graph and in the unrelated graph. If we find more than one component in the trace view graph, we find a parallel partition in the trace view graph. Analogously, if we find more than one component in the unrelated graph, we find a sequential partition in the trace view graph. In the given example (cf. Fig. 11), the unrelated graph contains more than one component; thus, we found a sequence cut. Afterward, we recursively continue on the partitions found, as exemplified in Fig. 5. Since either a parallel or a sequential partition can be found in general, only one of the two graphs contains more than one component. In short, computing maximal partitions is reduced to finding components in a graph, which can be solved in linear time [41].

The recursive partitioning described above, for example, consider Fig. 5 and Fig. 6b, yields a tree structure. Fig. 12 shows the tree structure of the high-level variant shown in Fig. 5 We exploit this tree structure to determine if two traces belong to the same variant—two traces belong to the same variant if their tree representations of the recursively partitioned high-level respectively low-level trace view are isomorphic. Note that checking if two trees are isomorphic can be done in polynomial time [42]. In summary,

Table 2
Overview of the event logs used for the evaluation.

Event log	#Traces	Timestamps	Bottom granularity
Hospital Billing [43]	100,000	start & completion	seconds
BPI Challenge 2012 (BPI Ch. 12) [18]	1,143	start & completion	milliseconds
BPI Challenge 2017 (BPI Ch. 17) [19]	1,050	start & completion	milliseconds
BPI Challenge 2018 (BPI Ch. 18) [44]	150,370	completion	milliseconds

we reduce the problem of computing variants to two well-studied graph problems, i.e., component detection and tree isomorphism. For a complete insight into the implementation, we refer to the publicly available source code of Cortado [39].

6.2. Visualizing variants

This section covers the visualization of the variants in the user interface of Cortado. Fig. 13a shows screenshots of Cortados's variant explorer that allows analysts to explore the variants visually. Basic statistics are shown next to each high-level variant: the absolute and relative number of traces corresponding to the displayed high-level variant and the number of corresponding low-level variants. Clicking on the number of low-level variants opens a new tab, showing all related low-level variants for the given high-level one; Fig. 13b shows an example. For the visualized real-life event logs in Fig. 13, we observe complex patterns similar to the running example, cf. Fig. 5.

7. Evaluation

This section presents an evaluation of the proposed variants. Sect. 7.1 presents automated experiments to demonstrate computational efficiency and applicability to real-world event logs. In addition, the automated experiments provide insight into the number of variants, the magnitude difference between low-level and high-level variants, and the effects of time granularity modifiers. Sect. 7.2 presents a user study assessing how high-level variants are used in event data analysis. The main goal of the user study is to assess the usefulness and ease of use of the proposed high-level variants compared to existing variant visualizations (cf. Fig. 1).

7.1. Automated experiments

This section presents the conducted automated experiments. The central goal of these experiments is to demonstrate that the proposed variants can be computed on real-life event logs within a reasonable time and thus demonstrate their practical applicability. Further, the automated experiments provide various insights into the number of variants and their dimensions regarding height and width for different time granularity modifiers.

Table 2 provides an overview of the event logs used. All event logs used [43,18,19,44] contain data of real-life processes, are publicly available, and contain partially ordered event data,[5] i.e., there exist traces containing activities with identical timestamps. Three out of four logs contain activities with heterogeneous temporal information; one log contains only time-point-based activities, i.e., BPI Ch. 18 (cf. Table 2). In the remainder of this section, we present the results and insights.

Table 3 shows the number of high-level and low-level variants for the different event logs and time granularities. According to Definitions 6 and 10, the number of high-level and low-level variants are identical for BPI Ch. 18, which only consists of single-timestamped activities. For the other logs, there are often more low-level than high-level variants. As discussed in Sect. 4, a one-to-many relation between high-level and low-level variants exists. The numbers in parentheses (cf. Table 3) indicate the share of variants that indicate parallel activities. We observe that the coarser the time granularity, the more variants indicate parallel process behavior. Overall, we observe that, for the selected event logs, a large share of variants indicate parallel process behavior, i.e., all values are clearly above 90% for all logs and time granularities.

Table 4 shows the time spent in seconds for calculating high-level and low-level variants from the given event logs per time granularity modifier. We can see that calculating high-level variants takes significantly longer than low-level variants across all logs. This observation can be explained by the fact that the recursion level for low-level variants is at most two levels deep because all elements represent time points, cf. Fig. 6b. Either 1) every element is parallel to each other, and thus only one recursion level is needed (i.e., a parallel partition), 2) all elements are sequentially ordered (i.e., a sequence partition), or 3) first a sequence partition and subsequently a parallel partition is applied. Compared to high-level variants, where elements may represent time points and time intervals, the partial order (i.e., Definition 6) may have more complex patterns leading to more recursion levels needed, for example, consider Fig. 5. For most logs, we observe the trend that the computation time usually increases with a coarser time granularity, cf. Table 4. Two primary causes can explain this observation. First, the more variants are computed, the longer the overall computation time. Second, the coarser the time granularity, the longer it takes to compute a single variant. Our implementation applies time granularity modifiers by adjusting the timestamps in real-time to avoid making copies of the event log with pre-modified timestamps. For instance, if the hours time granularity modifier is chosen, the milliseconds, seconds, and minutes

[5] We applied an event abstraction technique [45] on the Hospital Billing event log to obtain activities with start and completion timestamps.

Table 3
Total number of variants for different logs and time granularities. The number in parentheses below indicates the proportion of variants that contain parallel process behavior, i.e., at least one parallel partition is found in the corresponding high-level/low-level trace view.

Event log	Variant type	Time granularity					
		ms	s	m	h	days	month
Hospital Billing	high-level	-	667 (93.9%)	1,071 (97.4%)	1,048 (98.1%)	1,182 (98.4%)	1,389 (99.0%)
	low-level	-	685 (93.9%)	1,308 (97.1%)	1,331 (97.9%)	1,516 (98.2%)	1,773 (98.9%)
BPI Ch. 12	high-level	3,830 (98.8%)	3,766 (99.6%)	4,594 (100%)	5,220 (100%)	5,241 (100%)	4,080 (100%)
	low-level	3,855 (98.9%)	3,947 (99.7%)	5,737 (100%)	5,702 (100%)	5,257 (100%)	4,080 (100%)
BPI Ch. 17	high-level	5,937 (88.9%)	8,484 (100%)	9,665 (100%)	8,516 (100%)	9,454 (100%)	6,551 (100%)
	low-level	5,946 (88.9%)	9,137 (100%)	10,088 (100%)	8,995 (100%)	9,752 (100%)	6,551 (100%)
BPI Ch. 18	high-level	30,122 (100%)	33,407 (100%)	35,396 (100%)	29,313 (100%)	29,476 (100%)	28,294 (100%)
	low-level	30,122 (100%)	33,407 (100%)	35,396 (100%)	29,313 (100%)	29,476 (100%)	28,294 (100%)

Table 4
Calculation time (seconds) of variants for different logs and time granularities.

Event log	Variant type	Time granularity					
		ms	s	m	h	days	month
Hospital Billing	high-level	-	25	26	26	25	22
	low-level	-	18	18	17	13	10
BPI Ch. 12	high-level	26	28	29	32	35	41
	low-level	4	4	5	5	5	6
BPI Ch. 17	high-level	38	50	61	59	65	63
	low-level	10	12	13	13	14	14
BPI Ch. 18	high-level	673	771	900	825	900	1,180
	low-level	61	80	84	77	77	85

values are temporarily set to 0 when computing the variant for a particular trace. The specified times in Table 4 also include the time of these timestamp adjustments. The coarser the time granularity, the more values must be set to 0, which results in longer calculation times for variants with coarser time granularity.

Table 5 confirms the expected results that the variants' width decreases and their height increases when moving to coarser time granularities. Further, we observe that the average height of high-level variants is greater or equal to that of low-level variants.

Finally, Table 6 presents the share of high-level variants that contain non-singleton partitions as discussed in Sect. 4.4. As expected, we do not observe this phenomenon for BPI Ch. 18 since this log contains only atomic activities; thus, patterns, as exemplified in Fig. 7, cannot occur. For the other logs, we find that, in general, only a small number of variants are affected by the phenomenon that when sequence and parallel partitions are applied recursively, non-singletons remain.

7.2. User evaluation

This section presents the results of the conducted user study. The user study aims to assess how high-level variants support users in event data analysis tasks compared to existing visualizations, cf. Fig. 1 based on whether users perceive it *useful* and *easy to use* [46]. Further, the study aims to evaluate how the proposed high-level variant visualization supports users in event data analysis tasks compared to existing variant visualizations. Finally, with this study, we aim to gather open feedback on the positive and negative aspects of the proposed variant visualization as perceived by the participants. In Sect. 7.2.1, we present the study's design and execution and provide details about the participants. Sect. 7.2.2 discusses the results, and Sect. 7.2.3 threats to validity.

7.2.1. Study design, execution and participants

For the design of the user study, we consider three visualizations: (A) visualization of variants consisting of totally-ordered, time-point-based process activities, (B) visualization of variants consisting of partially ordered, time-point-based activities [15], and (C) the high-level variant visualization proposed in this paper (cf. Fig. 1).

Table 5
Average width and height (rounded to integers), i.e., the number of chevrons/bars respectively dots, of high-level and low-level variants for different logs and time granularities.

Dimension	Event log	Variant type	Time granularity					
			ms	s	m	h	days	month
width	Hospital Billing	high-level	-	6	6	5	5	3
		low-level	-	10	8	7	6	4
	BPI Ch. 12	high-level	20	18	13	10	6	2
		low-level	36	33	22	10	6	2
	BPI Ch. 17	high-level	19	13	9	6	5	2
		low-level	23	15	10	7	5	2
	BPI Ch. 18	high-level	58	52	36	29	25	9
		low-level	58	52	36	29	25	9
height	Hospital Billing	high-level	-	3	3	4	4	5
		low-level	-	2	3	3	4	5
	BPI Ch. 12	high-level	3	4	6	8	10	19
		low-level	3	4	5	7	10	19
	BPI Ch. 17	high-level	3	5	5	8	8	15
		low-level	2	3	4	7	8	15
	BPI Ch. 18	high-level	3	4	6	8	10	21
		low-level	3	4	6	8	10	21

Table 6
Share of high-level variants that contain non-singleton partitions, e.g., see Fig. 7.

Event log	Time granularity					
	ms	s	m	h	days	month
Hospital Billing	-	13%	4%	3%	2%	3%
BPI Ch. 12	0%	0%	14%	1%	0%	0%
BPI Ch. 17	7%	17%	14%	13%	10%	0%
BPI Ch. 18	0%	0%	0%	0%	0%	0%

Fig. 14 shows an example of an event log whose variants are visualized using the three techniques. In the user study, the variants displayed are extracted from real-life event logs, making the activity patterns realistic. We chose logs containing partially ordered activities with heterogeneous temporal information and decided to keep an amenable amount of variants shown to the user to avoid overwhelming the participants. Indeed, while piloting our design with several participants, we learned that they already found comparing small numbers of variants demanding, especially for variant visualization (A), cf. Fig. 14.

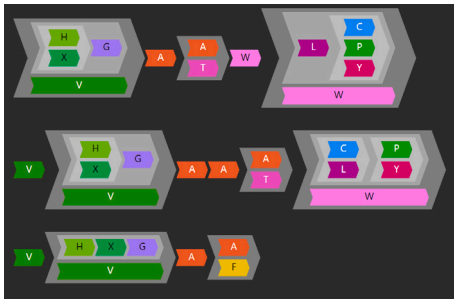
The study materials included a video tutorial and a questionnaire comprising open and closed questions organized follows.

- *Demographic questions*, aimed to collect participants' demographics and to screen them for their process mining experience.
- *Video tutorial*, introducing the visualizations (A), (B), and (C), cf. Fig. 14.
- *Comprehension & familiarization task*, aimed to check the participants' understanding of the three visualizations and allowing participants to familiarize with the kinds of questions asked in the following two tasks. This task was also planned to serve as a quality check for us to see if participants had understood the visualizations.
- *Task (1) "Extracting Patterns from Variant Visualizations"*, asking control-flow-related questions about activities.
  Overall, we planned to ask five data analysis questions per visualization, i.e., 15 questions in total
- *Task (2) "Identifying Variants based on Patterns"*, identifying variants that contain activity patterns described in natural language.
  Overall, we planned to ask five data analysis questions per visualization, i.e., 15 questions in total
- *Perceived usefulness* and *ease of use* questions about the visualizations.
- *Open questions*, aimed to collect qualitative feedback about (C).

We designed the questions of tasks (1) and (2) to mimic real-world use cases. Task (1) mimics a data exploration task. Given variant visualizations and patterns in natural language, participants had to identify true statements about whether the given patterns occurred in one of the visualized variants. Task (2) mimics a data filtering task. Given variant visualizations and patterns in natural language, participants had to select the variants satisfying these patterns. Each task includes all three visualizations and consists of five questions; we ask the same five questions per visualization. Within a task, we used the same event log for all visualizations, i.e., subsets of real-life event logs. We used letters to label activities in the log to mitigate the influence of domain knowledge. We also applied label randomization such that the questions per visualization differ, cf. Fig. 14, and randomized the answer options across different questions. This way, we aimed to reduce the learning effect and ensure fairness between the three visualizations since we always asked for the same patterns. After each task, we also asked participants to indicate the perceived task difficulty

(a) Variant visualization (A). Each chevron represents an atomic event. The suffix '+s' represents the start and '+c' the completion.

(b) Variant visualization (B). Each chevron represents an atomic event. The suffix '+s' represents the start and '+c' the completion.



(c) Variant visualization (C), i.e., the proposed high-level variants.

Fig. 14. Variant visualizations (A), (B), and (C) of an event log with heterogeneous temporal information used in the user study. We randomly changed the activity labels for each visualization; apart from the different labels, the shown variants describe the same event log.

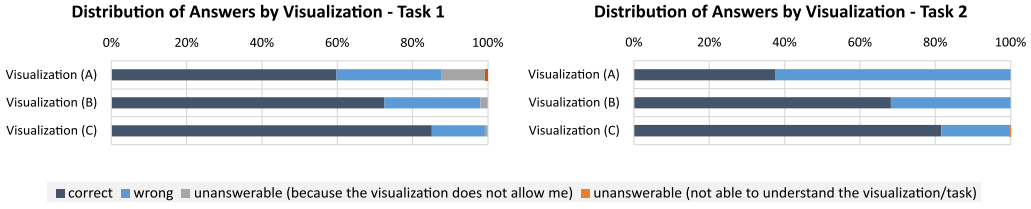


Fig. 15. Average distribution of accurate responses based on the questions per visualization for tasks (1) and (2). Participants tend to answer most questions correctly when they use (C).

per visualization to estimate the required effort. Regarding usefulness and ease of use, we designed the questionnaire following the validated question items presented in [46]. Finally, we asked participants to provide open feedback on visualization (C), and list positive and negative aspects.

The study was conducted in the summer of 2022 as an online questionnaire. We invited computer and data science students who took the courses "Business Process Intelligence" or "Advanced Process Mining" offered at RWTH Aachen University. We also invited researchers and industry professionals working with process mining. In total, 58 participants filled out the questionnaire. Based on the results of the *comprehension & familiarization task*, we filtered out seven responses that indicated low comprehension of the three variant visualizations. Of the remaining 51 responses, 32 are from bachelor/master students, 15 are Ph.D. students, two are postdocs/professors, and one works in the industry. All participants reported having expertise in process mining. On a scale of 1 (no expertise) to 6 (excellent expertise), the average expertise reported is 4.1, and no participant chose 1.

7.2.2. Results

Fig. 15 shows the distribution of correct answers per task and visualization. Overall, participants made fewer mistakes when using visualization (C) compared to (B) and (A). We further observe that, on average, across all five questions, about 10% indicated that visualization (A) did not allow them to answer the questions asked within task (1).

Fig. 16 shows the perceived task difficulty for each visualization. We observe that participants perceive the task difficulty as lowest when provided with visualization (C). Remember that we ask the same questions for each visualization.

Fig. 17 shows the *usefulness* and *ease of use* of the visualizations [46]. We observe that (C) has the highest scores both for usefulness and ease of use, followed by (B) and finally (A). Especially for usefulness, (C) has significantly better scores compared to the other visualizations, cf. the share of "extremely likely".

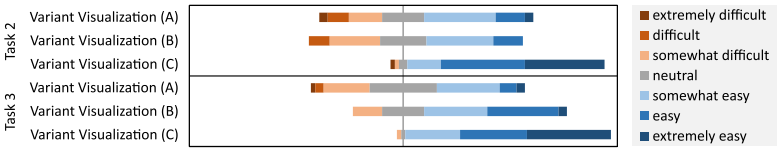
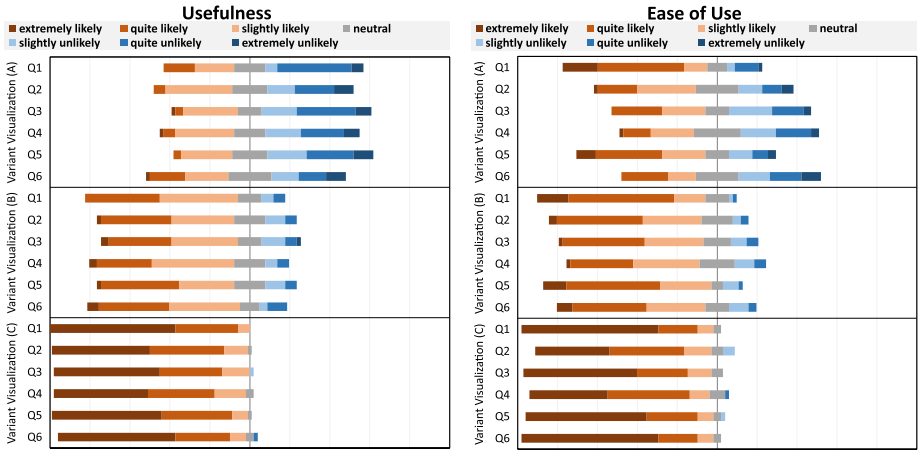


Fig. 16. Perceived task difficulty per visualization and task.



Q1 Using the visualization in my analysis would enable me to accomplish my analysis job more quickly.
Q2 Using the visualization would improve my performance in conducting my analysis tasks.
Q3 Using the visualization in my analysis job would increase my productivity.
Q4 Using the visualization would enhance my effectiveness on the analysis job.
Q5 Using the visualization would make it easier to do my analysis job.
Q6 I would find the visualization useful for my analysis job.

Q1 Learning to operate the visualization would be easy for me.
Q2 I would find it easy to get the visualization do what I want it to do.
Q3 My interaction with the visualization would be clear and understandable.
Q4 I would find the visualization to be flexible to interact with.
Q5 It would be easy for me to become skillful at using the visualization.
Q6 I would find the visualization easy to use.

Fig. 17. Perceived usefulness and ease of use.

Finally, we present the results from the open feedback. We received 49 responses on positive aspects related to (C), of which we summarize the most frequently mentioned. The participants indicated that visualization (C) is easy to interpret and compactly presents variants. They also reported that (C) enabled them to see parallelism within variants easily and allowed them to be noticeably faster at completing the tasks than the other visualizations. Regarding negative aspects, we collected 47 responses. Participants reported that the abstraction level chosen for high-level variants might be too high in some scenarios where one is interested in the exact overlap of activities. Further, they wrote that (C) does not allow one to see if activities are atomic or interval-based. We find these two mentioned negative aspects interesting as these are both covered by the proposed low-level variants (cf. Sect. 4.3) that were not part of the user study, as explained below. Finally, participants reported concerns about the vertical size of the variants, which can become considerably large in case many activities are executed in parallel. This concern motivates our interest in future research on variant clustering and simplification, as outlined in Sect. 8. To conclude, from our study, we can manifest that the proposed visualization (C) assists process analysts in data analysis better than existing variants as it is perceived as more useful and easier to use in the presence of parallel activities and seems to lead to fewer incorrect conclusions.

7.2.3. Study limitations and threats to validity

The results of the study should be seen in light of some limitations. First, the study was designed to evaluate high-level variants. As such all the questions posed were meant to be answerable with high-level variants and, as a consequence, they referred to the

level of abstraction chosen for them. Further, the event logs used do not contain patterns of chained activity instances, as exemplified in Fig. 7.

In addition, we requested participants to complete the questionnaire in one go, the study was unsupervised and participants could work on the questionnaire freely. Thus, some participants might not have completed all of the questionnaire at once and may not have remembered all the details when answering the questions on usefulness and ease of use, located at the questionnaire's end.

Regarding the generalizability of our results, we acknowledge that most participants are students, i.e., bachelor's, master's, and Ph.D. students. Although students can be suitable proxies for experts [47,48], generalizations from the questionnaire population to process mining experts, in general, should be considered limited. In addition, due to the limited number of study participants, the proposed study does not claim statistical significance. Thus, we recognize that further studies are needed to generalize our findings to a broader group of subjects. Still, since the core contribution of the paper is the definition and visualization of novel types of variants, we consider the limitations concerning generalization less critical.

Finally, we would like to remark that the study focused on comparing high-level variant visualizations with existing visualizations. Thus, this study did not evaluate other contributions presented in this paper, such as the low-level variants and the temporal granularity modifiers. We deliberately chose to exclude low-level variants from this study based on the results of our pilot. Indeed, while piloting the study, we discovered that answering the questions required a significant amount of time and concentration, taking about one to two hours to complete. To properly test low-level variants, participants should have responded to questions requiring at least a similar amount of time. Given that the study was designed as a voluntary online survey, there was a high risk that participants would manifest tiredness after some time or might drop out or not participate. To address and mitigate these risks, conducting a separate study to evaluate low-level variants is more advisable, as we plan for our future work.

The time granularity modifiers were also not evaluated in this user study. Indeed, to properly test the usefulness of time granularity modifiers, we require participants who understand the process captured in the log and can decide which modifier to apply based on this knowledge. In this scenario, an unsupervised questionnaire is not a suitable instrument. Instead, an observational study with expert analysts might provide more insights into how analysts use various time granularity modifiers during analysis and what variants they consider valuable for achieving specific analysis objectives.

8. Conclusion & future work

In this paper, we proposed two complementary variant definitions and corresponding visualizations, i.e., high-level and low-level variants, for partially ordered event data with heterogeneous temporal information. High-level and low-level variants enable the analysis of event data at different abstraction levels. Both proposed variant visualizations have been implemented in the open-source process mining tool Cortado. Further, we have discussed the importance of temporal granularity adjustments for variants and showed the impact on the variant calculation. The user study we performed showed promising results regarding the proposed high-level variant visualization, i.e., it seems more useful and easier to use than existing variant visualizations when analyzing event data with heterogeneous temporal information.

The variant definitions proposed in this paper open several directions for future work. For instance, clustering the proposed high- and low-level variants might be beneficial when dealing with large amounts of event data. For example, variants could be clustered based on the similarity of the order of their activities. Further, frequent pattern mining for the proposed variants is an interesting direction for future work as it facilitates users to oversee and explore large amounts of variants. Moreover, as previously discussed, further user studies are needed to evaluate particular aspects of our contribution, such as the usefulness of time granularity adjustments in variant visualizations for the analysis of event data or the interplay between low- and high-level variants in analysis tasks.

CRediT authorship contribution statement

Daniel Schuster: Conceptualization, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Francesca Zerbato:** Investigation, Writing – review & editing. **Sebastiaan J. van Zelst:** Supervision, Writing – review & editing. **Wil M.P. van der Aalst:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] W.M.P. van der Aalst, Process Mining: Data Science in Action, Springer, 2016, https://doi.org/10.1007/978-3-662-49851-4.

[2] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, F.M. Maggi, A. Marrella, M. Mecella, A. Soo, Automated discovery of process models from event logs: review and benchmark, IEEE Trans. Knowl. Data Eng. 31 (4) (2019) 686–705, https://doi.org/10.1109/TKDE.2018.2841877.

[3] B.F. van Dongen, A.K. Alves de Medeiros, L. Wen, Process mining: overview and outlook of Petri net discovery algorithms, in: K. Jensen, W.M.P. van der Aalst (Eds.), Transactions on Petri Nets and Other Models of Concurrency II, in: Lecture Notes in Computer Science, vol. 5460, Springer, 2009, pp. 225–242, https://doi.org/10.1007/978-3-642-00899-3_13.
[4] J. Carmona, B.F. van Dongen, A. Solti, M. Weidlich, Conformance Checking, Springer, 2018, https://doi.org/10.1007/978-3-319-99414-7.
[5] F. Caron, J. Vanthienen, B. Baesens, Comprehensive rule-based compliance checking and risk management with process mining, Decis. Support Syst. 54 (3) (2013) 1357–1369, https://doi.org/10.1016/j.dss.2012.12.012.
[6] W.L.J. Lee, H.M.W. Verbeek, J. Munoz-Gama, W.M.P. van der Aalst, M. Sepúlveda, Recomposing conformance: closing the circle on decomposed alignment-based conformance checking in process mining, Inf. Sci. 466 (2018) 55–91, https://doi.org/10.1016/j.ins.2018.07.026.
[7] G. Park, M. Song, Predicting performances in business processes using deep neural networks, Decis. Support Syst. 129 (2020) 113191, https://doi.org/10.1016/j.dss.2019.113191.
[8] M.T. Wynn, E. Poppe, J. Xu, A. ter Hofstede, R. Brown, A. Pini, W.M.P. van der Aalst, ProcessProfiler3D: a visualisation framework for log-based process performance comparison, Decis. Support Syst. 100 (2017) 93–108, https://doi.org/10.1016/j.dss.2017.04.004.
[9] D. Schuster, L. Schade, S.J. van Zelst, W.M.P. van der Aalst, Temporal performance analysis for block-structured process models in Cortado, in: J. de Weerdt, A. Polyvyanyy (Eds.), Intelligent Information Systems, in: Lecture Notes in Business Information Processing, vol. 452, Springer, 2022, pp. 110–119, https://doi.org/10.1007/978-3-031-07481-3_13.
[10] F. Taymouri, M. La Rosa, M. Dumas, F.M. Maggi, Business process variant analysis: survey and classification, Knowl.-Based Syst. 211 (2021) 106557, https://doi.org/10.1016/j.knosys.2020.106557.
[11] M.L. van Eck, X. Lu, S.J.J. Leemans, W.M.P. van der Aalst, PM^2: a process mining project methodology, in: J. Zdravkovic, M. Kirikova, P. Johannesson (Eds.), Advanced Information Systems Engineering, in: Lecture Notes in Computer Science, vol. 9097, Springer, 2015, pp. 297–313, https://doi.org/10.1007/978-3-319-19069-3_19.
[12] F. Zerbato, P. Soffer, B. Weber, Initial insights into exploratory process mining practices, in: A. Polyvyanyy, M.T. Wynn, A. van Looy, M. Reichert (Eds.), Business Process Management Forum, in: Lecture Notes in Business Information Processing, vol. 427, Springer, 2021, pp. 145–161, https://doi.org/10.1007/978-3-030-85440-9_9.
[13] H.L. Romero, R.M. Dijkman, P.W.P.J. Grefen, A.J. van Weele, Factors that determine the extent of business process standardization and the subsequent effect on business performance, Bus. Inf. Syst. Eng. 57 (4) (2015) 261–270, https://doi.org/10.1007/s12599-015-0386-0.
[14] W.M.P. van der Aalst, Process mining: discovering and improving Spaghetti and Lasagna processes, in: 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2011, https://doi.org/10.1109/CIDM.2011.6129461.
[15] W.M.P. van der Aalst, L. Santos, May I take your order?, in: A. Marrella, B. Weber (Eds.), Business Process Management Workshops, in: Lecture Notes in Business Information Processing, vol. 436, Springer, 2022, pp. 99–110, https://doi.org/10.1007/978-3-030-94343-1_8.
[16] C.S. Jensen, R.T. Snodgrass, Time instant, in: L. Liu, M.T. Özsu (Eds.), Encyclopedia of Database Systems, Springer, 2009, p. 3112, https://doi.org/10.1007/978-0-387-39940-9_1516.
[17] C.S. Jensen, R.T. Snodgrass, Time interval, in: L. Liu, M.T. Özsu (Eds.), Encyclopedia of Database Systems, Springer, 2009, pp. 3112–3113, https://doi.org/10.1007/978-0-387-39940-9_1423.
[18] B.F. van Dongen, BPI challenge 2012 - event log, https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f.
[19] B.F. van Dongen, BPI challenge 2017 - event log, https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b.
[20] H. de Oliveira, V. Augusto, B. Jouaneton, L. Lamarsalle, M. Prodel, X. Xie, Optimal process mining of timed event logs, Inf. Sci. 528 (2020) 58–78, https://doi.org/10.1016/j.ins.2020.04.020.
[21] D. Schuster, L. Schade, S.J. van Zelst, W.M.P. van der Aalst, Visualizing trace variants from partially ordered event data, in: J. Munoz-Gama, X. Lu (Eds.), Process Mining Workshops, in: Lecture Notes in Business Information Processing, vol. 433, Springer, 2022, pp. 34–46, https://doi.org/10.1007/978-3-030-98581-3_3.
[22] D.A. Keim, Information visualization and visual data mining, IEEE Trans. Vis. Comput. Graph. 8 (1) (2002) 1–8, https://doi.org/10.1109/2945.981847.
[23] M. Dumas, M. La Rosa, J. Mendling, H.A. Reijers, Fundamentals of Business Process Management, Springer, 2018, https://doi.org/10.1007/978-3-662-56509-4.
[24] K.M. van Hee, N. Sidorova, J.M. van der Werf, Business process modeling using Petri nets, in: D. Hutchison, T. Kanade, J. Kittler, J.M. Kleinberg, F. Mattern, J.C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M.Y. Vardi, G. Weikum, K. Jensen, W.M.P. van der Aalst, G. Balbo, M. Koutny, K. Wolf (Eds.), Transactions on Petri Nets and Other Models of Concurrency VII, in: Lecture Notes in Computer Science, vol. 7480, Springer, 2013, pp. 116–161, https://doi.org/10.1007/978-3-642-38143-0_4.
[25] S.J. Leemans, E. Poppe, M.T. Wynn, Directly follows-based process mining: exploration & a case study, in: 2019 International Conference on Process Mining (ICPM), IEEE, 2019, pp. 25–32, https://doi.org/10.1109/ICPM.2019.00015.
[26] W. Aigner, S. Miksch, H. Schumann, C. Tominski, Visualization of Time-Oriented Data, Springer, 2011, https://doi.org/10.1007/978-0-85729-079-3.
[27] T. Gschwandtner, Visual analytics meets process mining: challenges and opportunities, in: P. Ceravolo, S. Rinderle-Ma (Eds.), Data-Driven Process Discovery and Analysis, in: Lecture Notes in Business Information Processing, vol. 244, Springer, 2017, pp. 142–154, https://doi.org/10.1007/978-3-319-53435-0_7.
[28] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, G. Melançon, Visual analytics: definition, process, and challenges, in: A. Kerren, J.T. Stasko, J.-D. Fekete, C. North (Eds.), Information Visualization, in: Lecture Notes in Computer Science, vol. 4950, Springer, 2008, pp. 154–175, https://doi.org/10.1007/978-3-540-70956-5_7.
[29] M. Song, W.M.P. van der Aalst, Supporting process mining by showing events at a glance, in: Proceedings of the 17th Annual Workshop on Information Technologies and Systems (WITS), 2007, pp. 139–145.
[30] V. Denisov, D. Fahland, W.M.P. van der Aalst, Unbiased, fine-grained description of processes performance from event data, in: M. Weske, M. Montali, I. Weber, J. vom Brocke (Eds.), Business Process Management, in: Lecture Notes in Computer Science, vol. 11080, Springer, 2018, pp. 139–157, https://doi.org/10.1007/978-3-319-98648-7_9.
[31] T.R. Neubauer, G. Pamponet Sobrinho, M. Fantinato, S.M. Peres, Visualization for enabling human-in-the-loop in trace clustering-based process mining tasks, in: 2021 IEEE International Conference on Big Data (Big Data), IEEE, 2021, pp. 3548–3556, https://doi.org/10.1109/BigData52589.2021.9671985.
[32] M. Song, C.W. Günther, W.M.P. van der Aalst, Trace clustering in process mining, in: D. Ardagna, M. Mecella, J. Yang (Eds.), Business Process Management Workshops, in: Lecture Notes in Business Information Processing, vol. 17, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 109–120, https://doi.org/10.1007/978-3-642-00328-8_11.
[33] R.P.J.C. Bose, W.M.P. van der Aalst, Context aware trace clustering: towards improving process mining results, in: C. Apte, H. Park, K. Wang, M.J. Zaki (Eds.), Proceedings of the 2009 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2009, pp. 401–412, https://doi.org/10.1137/1.9781611972795.35.
[34] A. Yeshchenko, C. Di Ciccio, J. Mendling, A. Polyvyanyy, Visual drift detection for event sequence data of business processes, IEEE Trans. Vis. Comput. Graph. 28 (8) (2022) 3050–3068, https://doi.org/10.1109/TVCG.2021.3050071.
[35] B.F. van Dongen, A.K.A. de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, W.M.P. van der Aalst, The prom framework: a new era in process mining tool support, in: D. Hutchison, T. Kanade, J. Kittler, J.M. Kleinberg, F. Mattern, J.C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M.Y. Vardi, G. Weikum, G. Ciardo, P. Darondeau (Eds.), Applications and Theory of Petri Nets, in: Lecture Notes in Computer Science, vol. 3536, Springer, 2005, pp. 444–454, https://doi.org/10.1007/11494744_25.
[36] H. van der Aa, H. Leopold, M. Weidlich, Partial order resolution of event logs for process conformance checking, Decis. Support Syst. 136 (2020) 113347, https://doi.org/10.1016/j.dss.2020.113347.

[37] Y. Guo, S. Guo, Z. Jin, S. Kaul, D. Gotz, N. Cao, A survey on visual analysis of event sequence data, in: IEEE Transactions on Visualization and Computer Graphics PP, 2021, https://doi.org/10.1109/TVCG.2021.3100413.
[38] J.F. Allen, Maintaining knowledge about temporal intervals, Commun. ACM 26 (11) (1983) 832–843, https://doi.org/10.1145/182.358434.
[39] D. Schuster, S.J. van Zelst, W.M.P. van der Aalst, Cortado: a dedicated process mining tool for interactive process discovery, SoftwareX 22 (2023), https://doi.org/10.1016/j.softx.2023.101373.
[40] D. Schuster, S.J. van Zelst, W.M.P. van der Aalst, Utilizing domain knowledge in data-driven process discovery: a literature review, Comput. Ind. 137 (2022), https://doi.org/10.1016/j.compind.2022.103612.
[41] J. Hopcroft, R. Tarjan, Algorithm 447: efficient algorithms for graph manipulation, Commun. ACM 16 (6) (1973) 372–378, https://doi.org/10.1145/362248.362272.
[42] D.M. Campbell, D. Radford, Tree isomorphism algorithms: speed vs. clarity, Math. Mag. 64 (4) (1991) 252–261, https://doi.org/10.1080/0025570X.1991.11977616.
[43] Felix Mannhardt, Hospital billing - event log, https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741.
[44] B.F. van Dongen, F. Borchert, BPI challenge 2018 - event log, https://doi.org/10.4121/uuid:3301445f-95e8-4ff0-98a4-901f1f204972.
[45] F. Mannhardt, M. de Leoni, H.A. Reijers, W.M.P. van der Aalst, P.J. Toussaint, From low-level events to activities - a pattern-based approach, in: M. La Rosa, P. Loos, O. Pastor (Eds.), Business Process Management, in: Lecture Notes in Computer Science, vol. 9850, Springer, 2016, pp. 125–141, https://doi.org/10.1007/978-3-319-45348-4_8.
[46] F.D. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS Q. 13 (3) (1989) 319–340, https://doi.org/10.2307/249008.
[47] D. Falessi, N. Juristo, C. Wohlin, B. Turhan, J. Münch, A. Jedlitschka, M. Oivo, Empirical software engineering experts on the use of students and professionals in experiments, Empir. Softw. Eng. 23 (1) (2018) 452–489, https://doi.org/10.1007/s10664-017-9523-3.
[48] M. Svahnberg, A. Aurum, C. Wohlin, Using students as subjects - an empirical evaluation, in: D. Rombach, S. Elbaum, J. Münch (Eds.), Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM'08, ACM, New York, New York, USA, 2008, pp. 288–290, https://doi.org/10.1145/1414004.1414055.