Process Cubes: Slicing, Dicing, Rolling Up and Drilling Down Event Data for Process Mining

Wil M.P. van der Aalst

Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands;
Business Process Management Discipline, Queensland University of Technology, Brisbane, Australia; and
International Laboratory of Process-Aware Information Systems, National Research University Higher School of Economics, Moscow, Russia.
w.m.p.v.d.aalst@tue.nl

Abstract. Recent breakthroughs in *process mining* research make it possible to discover, analyze, and improve business processes based on event data. The growth of event data provides many opportunities but also imposes new challenges. Process mining is typically done for an isolated well-defined process in steady-state. However, the boundaries of a process may be fluid and there is a need to continuously view event data from different angles. This paper proposes the notion of *process cubes* where events and process models are organized using different dimensions. Each cell in the process cube corresponds to a set of events and can be used to discover a process model, to check conformance with respect to some process model, or to discover bottlenecks. The idea is related to the well-known OLAP (Online Analytical Processing) data cubes and associated operations such as slice, dice, roll-up, and drill-down. However, there are also significant differences because of the process-related nature of event data. For example, process discovery based on events is incomparable to computing the average or sum over a set of numerical values. Moreover, dimensions related to process instances (e.g. cases are split into gold and silver customers), subprocesses (e.g. acquisition versus delivery), organizational entities (e.g. backoffice versus frontoffice), and time (e.g., 2010, 2011, 2012, and 2013) are semantically different and it is challenging to slice, dice, roll-up, and drill-down process mining results efficiently.

Key words: OLAP, Process Mining, Big Data, Process Discovery, Conformance Checking

1 Introduction

Like most IT-related phenomena, also the growth of event data complies with Moore's Law. Similar to the number of transistors on chips, the capacity of hard disks, and the computing power of computers, the digital universe is growing exponentially and roughly doubling every 2 years [35, 40]. Although this is not a new phenomenon, suddenly many organizations realize that increasing amounts of "Big Data" (in the broadest sense of the word) need to be used intelligently in order to compete with other organizations in terms of efficiency, speed and service. However, the goal is not to collect as much data as possible. The real challenge is to turn event data into valuable insights. Only *process mining* techniques directly relate event data to end-to-end business processes [1]. Existing business process modeling approaches generating piles of process models are typically disconnected from the real processes and information systems. Data-oriented analysis techniques (e.g., data mining and machines learning) typically focus on simple classification, clustering, regression, or rule-learning problems.

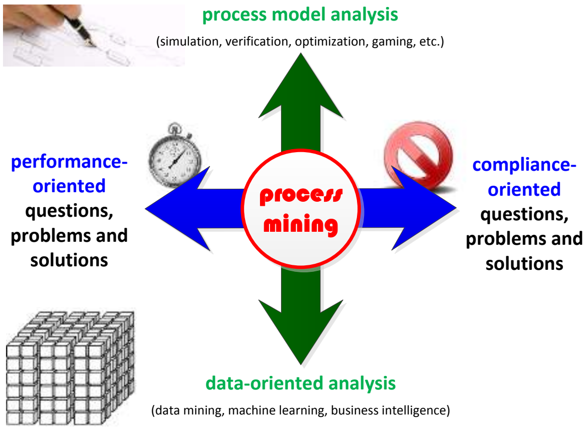
process model analysis
(simulation, verification, optimization, gaming, etc.)
performance-oriented questions, problems and solutions
process mining
compliance-oriented questions, problems and solutions
data-oriented analysis
(data mining, machine learning, business intelligence)

Fig. 1. Process mining provides the missing link between on the one hand process model analysis and data-oriented analysis and on the other hand performance and conformance.

Process mining aims to *discover, monitor and improve real processes by extracting knowledge from event logs* readily available in today's information systems [1]. Starting point for any process mining task is an *event log*. Each event in such a log refers to an *activity* (i.e., a well-defined step in some process) and is related to a particular *case* (i.e., a *process instance*). The events belonging to a case are *ordered* and can be seen as one "run" of the process, i.e., an event log can be viewed as a collection of *traces*. It is important to note that an event log contains only *example behavior*, i.e., we cannot assume that all possible traces have been observed [1].

The growing interest in process mining is illustrated by the *Process Mining Manifesto* [36] recently released by the *IEEE Task Force on Process Mining*. This manifesto is supported by 53 organizations and 77 process mining experts contributed to it.

The process mining spectrum is quite broad and includes techniques for process discovery, conformance checking, model repair, role discovery, bottleneck analysis, predicting the remaining flow time, and recommending next steps. Over the last decade hundreds of process mining techniques have been proposed. A process discovery technique uses as input an event log consisting of a collection of traces (i.e., sequences of events) and constructs a process model (Petri net, BPMN model, or similar) that "adequately" describes the observed behavior. A conformance checking technique uses as input an event log and a process model, and subsequently diagnoses differences between the observed behavior (i.e., traces in the event log) and the modeled behavior (i.e., possible runs of the model). Different process model notations can be used, e.g., BPMN models, BPEL specifications, UML activity diagrams, Statecharts, C-nets, or heuristic nets. MXML or XES (`www.xes-standard.org`) are two typical formats for storing event logs ready for process mining.

The incredible growth of event data poses new challenges [53]. As event logs grow, process mining techniques need to become more efficient and highly scalable. Dozens of process discovery [1, 11, 12, 16, 30, 18, 24, 25, 28, 31, 41, 54, 60, 61] and conformance checking [6, 13, 14, 15, 22, 29, 31, 42, 43, 51, 59] approaches have been proposed in literature. Despite the growing maturity of these approaches, the quality and efficiency of existing techniques leave much to be desired. State-of-the-art techniques still have problems dealing with large and/or complex event logs and process models.

Whereas traditional process mining techniques focus on the offline analysis of solitary processes in steady-state, this paper focuses on multiple inter-related processes that may change over time. Processes may change due to seasonal influences, working patterns, new laws, weather, and economic development. Moreover, there may be multiple variants of the same process or the process is composed of subprocesses. Existing techniques also cannot handle multiple process variants and/or heterogeneous collections of cases. However, in reality the same process may be used to handle very different cases, e.g., in a care process there may be characteristic groups of patients that need to be distinguished from one another. Moreover, there may be different variants of the same process, e.g., different hospitals execute similar care processes, and it is interesting to compare them. Obviously, it is very challenging to discover and compare processes for different hospitals and patient groups. Unfortunately, traditional techniques tend to focus on a single well-defined process. Cases can be clustered in groups and process models can be compared, however, *there are no process discovery techniques that produce overarching models able to relate and analyze different groups and process variants*. For example, we have applied process discovery in over 25 municipalities executing similar processes. However, there are no discovery approaches relating these process variants.

In this paper, we propose the new notion of *process cubes* where events and process models are organized using different dimensions (e.g., case types, event classes, and time windows). A *process cube* may have any number of dimensions used to distribute process models and event logs over multiple cells. The first process cube shown in Figure 2(left) has three dimensions: *case type*, *event class* and *time window*. In this simple example, there is only one case type and only

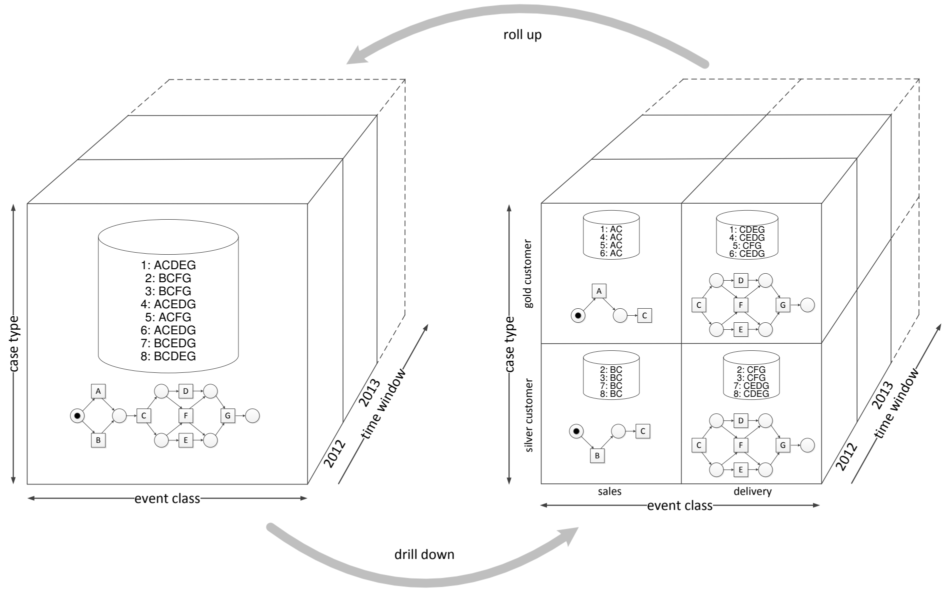

Fig. 2. Two process cubes illustrating the splitting (drilling down) and merging (rolling up) of process cells using the case type and event class dimensions.

one event class. The cube covers multiple time windows, but only one is shown (all cases completed in 2012). In this toy example there are only eight cases (i.e., process instances) and seven distinct activities. The process may be split by identifying multiple case types and/or multiple event classes. The second process cube shown on the right-hand side of Figure 2 has two case types (gold customer and silver customer) and two event classes (sales and delivery).

The *case type dimension* is based on properties of the case the event belongs to. In Figure 2(right), cases 1, 4, 5, and 6 refer to a “gold customer”. Hence, the cells in the “gold customer” row include events related to these four cases. The *event class dimension* is based on properties of individual events, e.g., the event’s activity name, its associated resource, or the geographic location associated with the event. In Figure 2(right), the event class dimension is based on the activity of each event. The event class “sales” includes activities A, B, and C. The event class “delivery” refers to activities C, D, E, F, and G. The *time window dimension* uses the timestamps found in the event log. A time window may refer to a particular day, week, month, or any other period.

Each cell in a process cube refers to a collection of events and possibly also process mining results (e.g., a discovered process model) or other artifacts (e.g., a set of business rules). Events may be included in multiple cells, e.g., sales and delivery cells share C events. Each of the three dimensions may have an associated hierarchy, e.g., years composed of months and months composed of days.

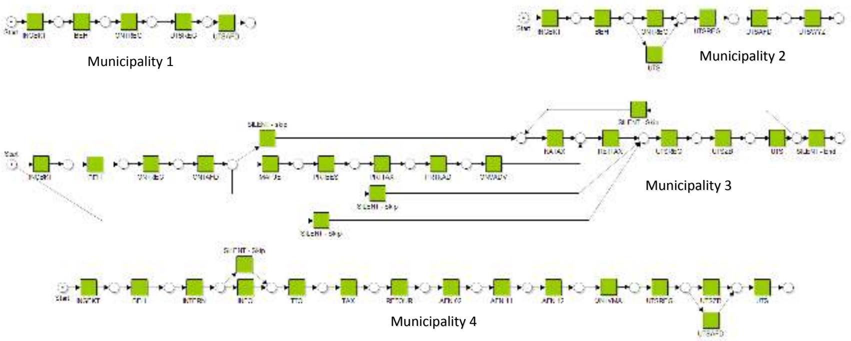

Fig. 3. Process models showing how complaints regarding the taxation of real estate are handled within four Dutch municipalities.

Figure 3 illustrates the relevance of process cubes using four variants of the same municipal complaints handling process. The process models in Figure 3 show that four of the ten municipalities involved in our CoSeloG project[1] are handling complaints related to the taxation of houses very differently [20]. For each of the four processes we have event data and normative process models. The average throughput times of the four process variants are very different, e.g., Municipality 1 handles complaints in 22 days whereas Municipality 3 uses 227 days. We aim at organizing such event data and process models in a process cube that allows for easy comparison of processes between municipalities, over time, and for different groups of citizens.

Process cubes are related to the well-known OLAP (Online Analytical Processing) cubes [27] and large process model repositories [49]. In an OLAP cube, one can drill-down or roll-up data, zoom into a selected slice of the overall data, or reorder the dimensions. However, OLAP cubes cannot be used for process-related data since events are ordered and belong to cases. Moreover, cells are associated to process models and not just event data. Conversely, process model repositories do not store event data. In process cubes, models and event data are directly related. Observed and modeled behavior can be compared, models can be discovered from event data, and event data can be used the breathe life into otherwise static process models.

This paper defines OLAP notions such as "slicing", "dicing", "rolling up" and "drilling down" for event data. These can be used to compare, merge, and split process cells at both the log and model level. The process cube notion is closely related to *divide-and-conquer approaches in process mining* where huge event logs are partitioned into smaller sublogs to improve performance and scalability. In principle, process cubes can also be used to decompose challenging process mining problems into smaller problems using the techniques described in [3, 5, 4]. These techniques may be used to speed-up OLAP operations.

[1] See the CoSeLoG (Configurable Services for Local Governments) project home page, `www.win.tue.nl/coselog`.

The remainder of this paper is organized as follows. Section 2 introduces the *process cube* notion and further motivates its relevance. Section 3 formalizes the *event base* used to create process cubes, i.e., the source information describing "raw" events and their properties. The so-called *process cube structure* is defined in Section 4. This structure defines the *dimensions* of the cube. Event base and process cube structure are linked through the so-called *process cube view* defined in Section 5. Section 6 defines the *slice* and *dice* operations on process cubes. *Roll-up* and *drill-down* are formalized in Section 7. Section 8 concludes the paper by discussing innovations and challenges.

2 Process Cubes

As illustrated by Figure 4, event data can be used to construct a *process cube*. Each *cell* in the process cube corresponds to a *set of events* selected based on the corresponding *dimension* values. In Figure 4 there are three dimensions. However, a process cube can have any number of dimensions $n \in \mathbb{N}$. Moreover, dimensions can be based on any event property. In Figure 4 events are grouped in cells based on *case type*, a particular *event class*, and a particular *time window*, i.e., one cell refers to the set of all events belonging to case type ct, event class ec, and time window tw. The *case type dimension* is based on properties of the case as a whole and not on the characteristics of individual events. Hence, if event e is of type ct, then all events of the case to which e belongs, also have type ct. Case type ct may be based on the type of customer (gold of silver) or on the total amount (e.g., < 1000 or ≥ 1000). The *event class dimension* is based on properties of the individual events, e.g., the event's activity name, associated resources, or geographic location. Event type (et) may depend on the activity name, e.g., there could be three event classes based on overlapping sets of activity names: $\{A, B\}$, $\{C, D\}$, and $\{E\}$. The *time window dimension* uses the timestamps found in the event log. A time window (tw) may refer to a particular day, week, month, or any other period, e.g., to all events that took place in December 2012.

An event may belong to multiple process cells because case types, event classes, and time windows may be overlapping. Process cells may be merged into larger cells, i.e., event data can be grouped at different levels of granularity. Semantically, the merging of cells corresponds to the merging of the corresponding event sets. One may refine or coarsen a dimension.

A *process cube* is composed of a set of process cells as shown in Figure 4. Per cell one may have a predefined or discovered process model. The process model may have been discovered from the cell's event data or given upfront. Moreover, other artifacts, e.g., organizational models [56], may be associated to individual cells.

Process cubes will be used to relate different processes, e.g., we may be interested in understanding the differences between gold and silver customers, large orders and small orders, December and January, John and Ann, etc. Moreover,

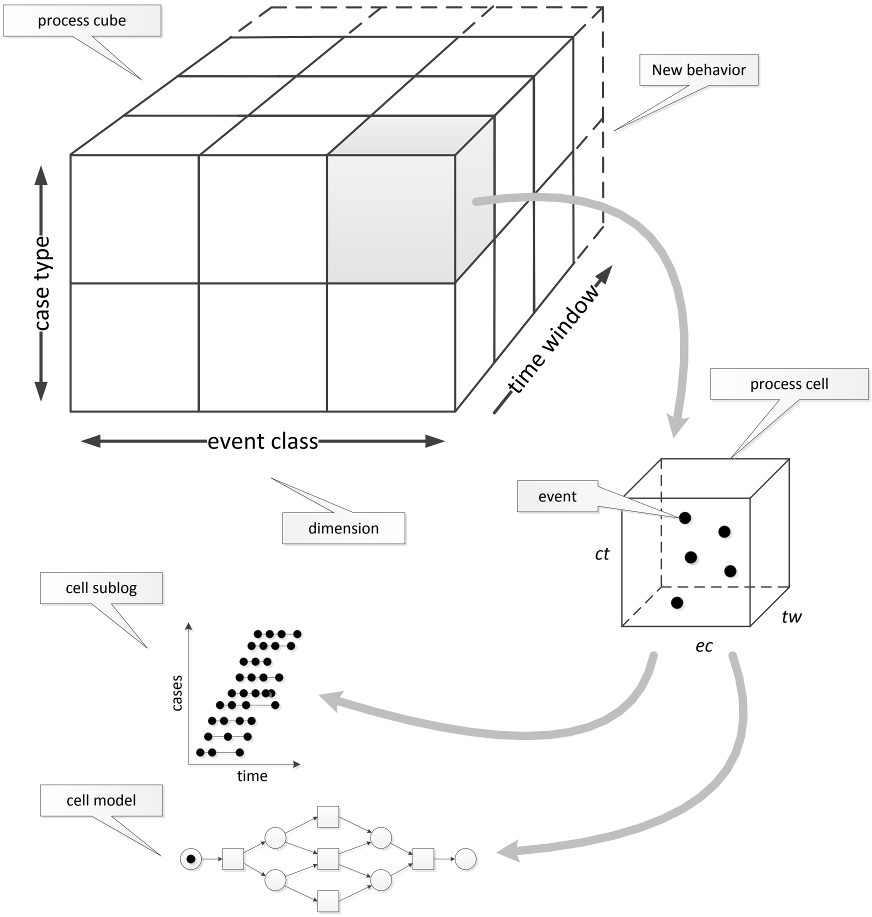

Fig. 4. A process cube relates events to different dimensions. Each cell in the cube corresponds to a sublog containing events and may have an associated set of models or other artifacts (derived or given as input).

we may want to chop a larger cell into many smaller cells for efficiency reasons (e.g., distributing a time-consuming discovery task).

The three dimensions shown in Figure 4 only serve as examples and may be refined further, e.g., there may be multiple dimensions based on various classifications of cases (e.g., customer type, region, size, etc.). Moreover, each dimension may have a natural hierarchical structure (e.g., a year is composed of months and a country is composed of regions) that can be exploited for the aggregation, refinement, and selection of event data.

Process cells (and the associated sublogs and models) can be split and merged in two ways as is illustrated in Figure 5. The *horizontal dimension* of a cell refers to model elements (typically activities) rather than cases. The *vertical dimension* of a cell refers to cases rather than model elements. Consider for

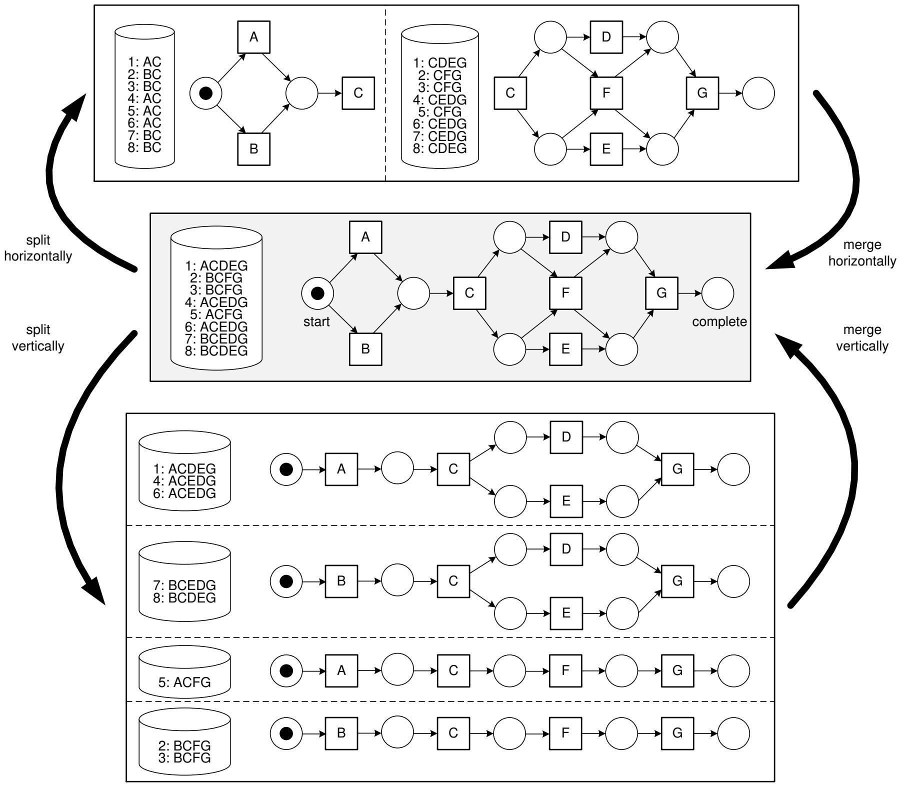

Fig. 5. Illustration of the two types of merging and splitting process cells.

example the event log depicted in the middle of Figure 5. The event log consists of 8 cases and 37 events. If the log is split horizontally based on the two overlapping event classes $\{A, B, C\}$ and $\{C, D, E, F, G\}$, then we obtain two sublogs each consisting of all 8 cases. The top-left part of Figure 5 shows the new process cell corresponding to event class $\{A, B, C\}$. Model and event log abstract from activities $\{D, E, F, G\}$. The top-right part of Figure 5 shows the process cell corresponding to event class $\{C, D, E, F, G\}$. Note that the cell's sublog and model abstract from activities A and B. If the log is split vertically, we could obtain the four sublogs depicted in the lower half of Figure 5. Each cell contains a subset of cases selected according to some criterion, e.g., the type of customer or the set of activities executed. Unlike the horizontal split, no individual events are removed, i.e., all events belonging to a case are included in the cell (or no events of the case are included).

The seven process models shown in Figure 5 may correspond to discovered or modeled behaviors. The models in the horizontally split cells are in a "part of" relationship, i.e., they are fragments of the larger model and cover only subsets of activities. The models in the vertically split cells are in an "is a" relationship, i.e., they can be viewed as specializations of original model covering only subsets of cases. *The case type and time window dimensions in Figure 4 are often used to merge or split a log vertically. The event class dimension is often used to merge or split a log horizontally.*

Obviously, there are some similarities between a process cube and an OLAP (Online Analytical Processing) cube [27]. In an OLAP cube, one can drill-down or roll-up data, zoom into a selected slice of the overall data, or reorder the dimensions. As shown in [46], these ideas can be applied to event data. Any selection of cells in the process cube can be used to materialize an event log and discover the corresponding process model. However, unlike [46] which focuses on a multi-dimensional variant of the well-know heuristic miner [60], we aim at a much more general approach. On the one hand, we allow for any number of dimensions and any process mining technique (not just discovery). On the other hand, we take into account the essential properties of event data and processes. For example, the two types of merging and splitting process cells illustrated by Figure 5 are process-specific and do not correspond to existing OLAP notions. Techniques for merging and splitting process cells are related to divide-and-conquer approaches for process mining (e.g., to distribute process discovery or conformance checking) [3].

Based on the ideas shown in Figure 4, we have developed an initial prototype (called *ProCube*) using the process mining framework *ProM* and the *Palo OLAP* toolset (*JPalo* client and *Palo* MOLAP server) [39]. *ProCube* application runs as a plugin in *ProM*. *Palo* is employed for its OLAP capabilities. The *ProCube* plugin creates sublogs per cell on-the-fly and visualizes process models discovered using the fuzzy [34] and heuristics [60] miner, social networks derived using ProM's social network miner [10], and dotted charts [55] computed per cell.

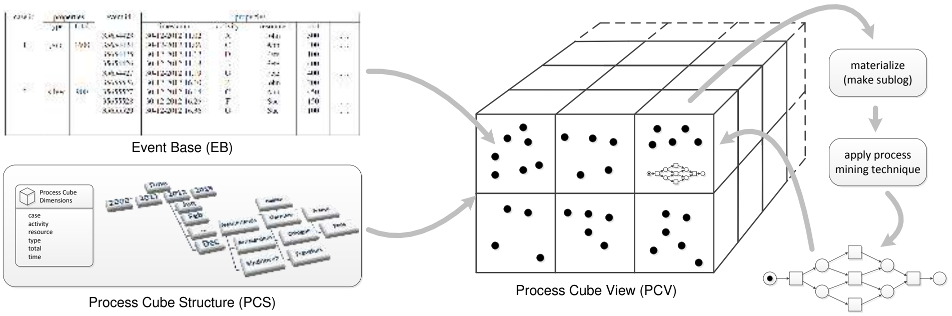

Fig. 6. Overview of the different ingredients needed to define and use process cubes.

In the remainder, we do not focus on specific process mining techniques or a specific implementation of the process cube notion. Instead, we conceptualize the essential ideas. Figure 6 lists the different ingredients described next. The *Event Base* (EB) contains information about actually recorded events (Section 3). These events may have different properties, some of which are used as dimensions in the *Process Cube Structure* (PCS) described in Section 4. A *Pro-*

case id	properties		event id	properties			
	type	total		timestamp	activity	resource	cost
1	gold	1600	35654423	30-12-2012:11.02	A	John	300
			35654424	30-12-2012:11.06	C	Ann	400
			35654425	30-12-2012:11.12	D	Pete	100
			35654426	30-12-2012:11.18	E	Pete	400
			35654427	30-12-2012:11.19	G	Pete	400
2	silver	900	35655526	30-12-2012:16.10	B	John	200
			35655527	30-12-2012:16.14	C	Ann	450
			35655528	30-12-2012:16.26	F	Sue	150
			35655529	30-12-2012:16.36	G	Sue	100
...	...	...	...	...	...	...	...

Table 1. A fragment of some event log: each line corresponds to an event.

cess Cube View (PCV) uses both EB and PCS to create a concrete view. The view may be modified using typical OLAP operations such as slice and dice (Section 6) and roll-up and drill-down (Section 7). Any process mining technique can be applied to a cell in the selected view. To do this, the cell's event data need to be materialized to create a sublog that is used as input by conventional process mining techniques. These techniques may produce (process) models, charts, etc. The results are stored per cell and the different cells can be compared systematically.

3 Event Base

Normally, *event logs* serve as the starting point for process mining. These logs are created having a particular process and a set of questions in mind. An event log can be viewed as a multiset of *traces*. Each trace describes the life-cycle of a particular *case* (i.e., a *process instance*) in terms of the *activities* executed. Often event logs store additional information about events. For example, many process mining techniques use extra information such as the *resource* (i.e., person or device) executing or initiating the activity, the *timestamp* of the event, or *data elements* recorded with the event (e.g., the size of an order). Table 1 shows a small fragment of some larger event log. Only two traces are shown. Each event has a unique id and several properties. For example, event 35654423 is an instance of activity A that occurred on December 30th at 11.02, was executed by John, and costs 300 euros. The second trace starts with event 35655526 and also refers to an instance of activity A. Note that each trace corresponds to a *case*, i.e., a completed process instance. Also cases may have properties as is shown in Table 1 where cases have a customer *type* (gold or silver) and total *amount*, e.g., case 1 was executed for a gold customer and had a total amount of 1600 euro. Implicitly, an event inherits the properties of the corresponding case.

For process cubes we consider an *event base*, i.e., a large collection of events not tailored towards a particular process or predefined set of questions. An event base can be seen as an all-encompassing event log or the union of a collection of related event logs.

Properties of events have values and the dimensions of a process cube structure sets of possible property values. Throughout the paper we assume the following universes.

Definition 1 (Universes). *$\mathcal{U}_V$ is the universe of possible attribute values (e.g., strings, numbers, etc.). $\mathcal{U}_S = \mathcal{P}(\mathcal{U}_V)$ is the universe of value sets. $\mathcal{U}_H = \mathcal{P}(\mathcal{U}_S)$ is the universe of value set collections (set of sets).*

Note that $v \in \mathcal{U}_V$ is a single value (e.g., $v = 300$), $V \in \mathcal{U}_S$ is a set of values (e.g., $V = \{gold, silver\}$), and $H \in \mathcal{U}_H$ is a collection of sets. For example, $H = \{\{a, b, c\}, \{c, d\}, \{d, e, f\}\}$ or $H = \{\{x \in \mathbb{N} \mid x < 50\}, \{x \in \mathbb{N} \mid 50 \leq x < 60\}, \{x \in \mathbb{N} \mid x \geq 60\}\}$.

Definition 2 (Event Base). *An* event base *$EB = (E, P, \pi)$ defines a set of* events *E, a set of* event properties *P, and a function $\pi \in P \rightarrow (E \not\rightarrow \mathcal{U}_V)$. For any property $p \in P$, $\pi(p)$ (denoted π_p) is a partial function mapping events onto values. If $\pi_p(e) = v$, then event $e \in E$ has a property $p \in P$ and the value of this property is $v \in \mathcal{U}_V$. If $e \notin dom(\pi_p)$, then event e does not have property p and we write $\pi_p(e) = \perp$ to indicate this.*

The set E refers to the individual events. For example event $e = 35654423$ in Table 1 may be such an event. Note that an event identifier $e \in E$ may be generated implicitly (it has no meaning). P is the set of properties that events may or may not have. For example, $P = \{case, activity, time, resource, cost, type, total\}$ corresponds to the columns in Table 1. $\pi_{case}(35654423) = 1$, $\pi_{activity}(35654423) = A$, and $\pi_{resource}(35654423) = John$ are some of the properties of the first event in Table 1. In the remainder we assume the standard properties *case*, *activity*, and *time* to be defined for all events, i.e., $dom(\pi_{case}) = dom(\pi_{activity}) = dom(\pi_{time}) = E$. For example, we do not allow for events not related to a case. An attribute like *resource* is optional. Note that π defines a partial function per property p and missing values are represented as $\pi_p(e) = \perp$. For example, if $\pi_{resource}(e) = \perp$, then $e \notin dom(\pi_{resource})$ implying that e does not have an associated resource.

Assume that $\mathcal{U}_A$ is the set of activities appearing in $EB = (E, P, \pi)$. Given a set of events $E' \subseteq E$, we can compute a multiset of traces $L \in (\mathcal{U}_A)^* \rightarrow \mathbb{N}$ where each trace $\sigma \in L$ corresponds to a case. For example, case 1 in Table 1 can be presented as $\langle A, C, D, E, G \rangle$ and case 2 as $\langle B, C, F, G \rangle$. Most control-flow discovery techniques [1, 11, 12, 16, 30, 18, 24, 25, 28, 31, 41, 54, 60, 61] use such a simple representation as input. This representation ignores concrete timestamps (only the order matters) and abstracts from properties such as *resource*, *cost*, *type*, and *total*.

Note that given an event base, one can derive additional properties. For example, we can take different event properties together, e.g., $\pi_{ar}(e) = (\pi_{activity}(e), \pi_{resource}(e))$. Such derived properties may also be based on other events. For example, $\pi_{st}(e) = min\{\pi_{time}(e') \mid e' \in E \ \wedge \ \pi_{case}(e) = \pi_{case}(e')\}$ is the start time of the case e belongs to, and $\pi_{sum}(e) = sum\{\pi_{costs}(e') \mid e' \in E \ \wedge \ \pi_{case}(e) = \pi_{case}(e')\}$ are the total costs of the case e belongs to. Many useful event attributes can be derived from information inside or outside the initial event base [9]. For example, one can estimate the "stress level" of a resource working on event e by computing the number of queueing activities. In the remainder we assume an event base $EB = (E, P, \pi)$ that includes all properties that may serve as dimensions of the process cube (including derived ones).

4 Process Cube Structure

Independent of the event base EB we define the *structure* of the process cube. The structure is fully characterized by the *dimensions* of the cube.

Definition 3 (Process Cube Structure). *A* process cube structure *is a triplet $PCS = (D, type, hier)$ where:*

- *D is a set of dimensions,*
- *$type \in D \to \mathcal{U}_S$ is a function defining the possible set of values for each dimension, e.g., $type(age) = \{0, 1, 2, \ldots, 120\}$ for $age \in D$, and*
- *$hier \in D \to \mathcal{U}_H$ defines a hierarchy for each dimension such that for any $d \in D$: $type(d) = \bigcup hier(d)$. Note that a hierarchy is merely a collection of sets of values.*

A dimension $d \in D$ has a type $type(d)$ and a hierarchy $hier(d)$. $type(d)$ is the set of possible values and typically only a fraction of these values are present in a concrete instance of the process cube. For example, $type(cost) = \mathbb{N}$ allows for infinitely many possible values.

A hierarchy $hier(d)$ is a set of sets. For example $hier(time)$ contains sets such as T_{2011}, T_{2012}, and T_{2013} each representing all possible timestamps in a particular year.[2] These sets do not need to be disjoint. For example, $hier(time)$ may also contain sets such as $T_{Dec-2012}$ (all possible timestamps in December 2012), $T_{Tue-2012}$ (all Tuesdays in 2012), and $T_{30-12-2012}$ (December 30th 2012). These sets may form a hierarchy based on set inclusion, for example T_{2012} dominates $T_{Dec-2012}$ because $T_{Dec-2012} \subseteq T_{2012}$. Sets may also be partially overlapping, e.g., $T_{Tue-2012} \cap T_{Dec-2012} \neq \emptyset$.

In order to relate an event base and a process cube structure both need to be *compatible*, i.e., dimensions should correspond to properties and concrete event property values need to be of the right type.

Definition 4 (Compatible). *A process cube structure $PCS = (D, type, hier)$ and an event base $EB = (E, P, \pi)$ are compatible if*

- *$D \subseteq P$, i.e., dimensions correspond to properties, and*
- *for any $d \in D$ and $e \in E$: $\pi_d(e) \in type(d)$.*

[2] Note that the notation T_X always refers to a set of timestamps meeting constraint X, e.g., $T_{30-12-2012}$ are all timestamps on the specified day.

There are different ways of dealing with *missing values*. The above definition allows for missing values if $\perp \in type(d)$. If $\perp \notin type(d)$, then compatibility implies $dom(\pi_d) = E$.

5 Process Cube View

While applying typical OLAP operations such as slice, dice, roll-up and drill-down the event base $EB = (E, P, \pi)$ and process cube structure $PCS = (D, type, hier)$ do not change. It is merely a change of the way event data is viewed. A *process cube view* defines which dimensions are visible and which events are selected.

Definition 5 (Process Cube View). *Let $PCS = (D, type, hier)$ be a process cube structure. A* process cube view *is a pair $PCV = (D_{sel}, sel)$ such that:*

- *$D_{sel} \subseteq D$ are the selected dimensions,*
- *$sel \in D \to \mathcal{U}_H$ is a function selecting the part of the hierarchy considered per dimension. Function sel is such that for any $d \in D$:*
  - *$sel(d) \subseteq hier(d)$, and*
  - *for any $V_1, V_2 \in sel(d)$: $V_1 \subseteq V_2$ implies $V_1 = V_2$.*

A process cube view defines a cube with $k = |D_{sel}|$ dimensions. The maximal number of dimensions is set by D, i.e., all dimensions defined in the process cube structure ($D_{sel} \subseteq D$). Function *sel* selects sets of values per dimension (including dimensions not selected in D_{sel}). For example, when slicing a cube one decision is removed, but the removed dimension is still used for filtering. Given a dimension $d \in D$, $sel(d)$ defines the elements on the d axis. For example, $sel(time) = \{T_{2011}, T_{2012}, T_{2013}\}$ states that the *time* dimension has three elements. This implies that events before 2011 are filtered out. Moreover, we do not distinguish events based on the month, day or time; only the year matters. $sel(time) = \{T_{2011}, T_{Jan-2012}, T_{Feb-2012}, \ldots, T_{Dec-2012}, T_{2013}\}$ is an alternative view for the *time* dimension. Now the different months of 2012 are distinguished. $sel(d) \subseteq hier(d)$ ensures that the elements of the d dimension are consistent with the process cube structure. The last requirement ($V_1 \subseteq V_2$ implies $V_1 = V_2$) implies that the elements of $sel(d)$ are non-dominating. For example, it would not make sense to have $sel(time) = \{T_{2012}, T_{Jan-2012}\}$ because $T_{Jan-2012} \subseteq T_{2012}$.

As shown in Figure 6, the process cube view can be used to create a *sublog* per cell in the process cube view based on the event base. These sublogs can be viewed as conventional event logs and any process mining technique can be applied to them.

Definition 6 (Materialized Process Cube View). *Let process cube structure $PCS = (D, type, hier)$ and event base $EB = (E, P, \pi)$ be compatible. The materialized process cube for some view $PCV = (D_{sel}, sel)$ of PCS is $M_{EB,PCV} = \{(c, events(c)) \mid c \in cells\}$ with $cells = \{c \in D_{sel} \to \mathcal{U}_S \mid \forall_{d \in D_{sel}}\ c(d) \in sel(d)\}$ being the cells of the cube and $events(c) = \{e \in E \mid \forall_{d \in D_{sel}}\ \pi_d(e) \in c(d)\ \wedge\ \forall_{d \in D}\ \pi_d(e) \in \bigcup sel(d)\}$ the set of events per cell.*

cells is the collection of cells of the cube. A $c \in \mathit{cells}$ is an assignment of each visible dimension to precisely one element of that dimension, e.g., $c(\mathit{time}) = T_{Jan-2012}$, $c(\mathit{resource}) = \{\mathit{John}, \mathit{Pete}\}$, and $c(\mathit{type}) = \{\mathit{gold}\}$. $\mathit{events}(c)$ are all events corresponding to cell c (first requirement: $\forall_{d \in D_{sel}}\ \pi_d(e) \in c(d)$) and not filtered out (second requirement: $\forall_{d \in D}\ \pi_d(e) \in \bigcup \mathit{sel}(d)$).

Definition 6 provides the interface to existing process discovery [1, 11, 12, 16, 30, 18, 24, 25, 28, 31, 41, 54, 60, 61] and conformance checking [6, 13, 14, 15, 22, 29, 31, 42, 43, 51, 59] techniques. $M_{EB,PCV}$ defines how to compute an event log (called sublog) per cell. As shown in Figure 6, these sublogs can be used to compute results per cell.

Note that the materialized process cube view $M_{EB,PCV}$ may be constructed on-the-fly or pre-computed. Existing OLAP tools often materialize views in order to enable interactive analysis. However, for process mining techniques it is typically not known how to do this efficiently.

6 Slice and Dice

Next we consider the classical OLAP operations in the context of our process cubes.

The *slice operation* produces a sliced OLAP cube by allowing the analyst to pick specific value for one of the dimensions. For example, for sales data one can slice the cube for location "Eindhoven", i.e., the location dimension is removed from the cube and only sales of the stores in Eindhoven are considered. Slicing the cube for the year "2012" implies removing the time dimension and only considering sales in 2012. The *dice operation* produces a subcube by allowing the analyst to pick specific values for multiple dimensions. For example, one could dice the sales OLAP cube for years "2012" and "2013" and locations "Eindhoven" and "Amsterdam". No dimensions are removed, but only sales in 2012 and 2013 in stores in Eindhoven and Amsterdam are considered.

Given the earlier formalizations, we can easily define the slice operation for process cubes.

Definition 7 (Slice). *Let* $PCS = (D, \mathit{type}, \mathit{hier})$ *be a process cube structure and* $PCV = (D_{sel}, \mathit{sel})$ *a view of* PCS*. For any* $d \in D_{sel}$ *and* $V \in \mathit{sel}(d)$*:* $\mathit{slice}_{d,V}(PCV) = (D'_{sel}, \mathit{sel}')$ *with* $D'_{sel} = D_{sel} \setminus \{d\}$*,* $\mathit{sel}'(d) = \{V\}$*, and* $\mathit{sel}'(d') = \mathit{sel}(d')$ *for* $d' \in D \setminus \{d\}$*.*

$\mathit{slice}_{d,V}(PCV)$ produces a new process cube view. Note that d is no longer a visible dimension: $d \notin D'_{sel}$. At the same time d is used to filter events: only events e with $\pi_d(e) \in V$ are considered in the new view.

Definition 8 (Dice). *Let* $PCS = (D, \mathit{type}, \mathit{hier})$ *be a process cube structure and* $PCV = (D_{sel}, \mathit{sel})$ *a view of* PCS*. Let* $\mathit{res} \in D_{sel} \not\rightarrow \mathcal{U}_H$ *be a restriction such for any* $d \in \mathit{dom}(\mathit{res})$*:* $\mathit{res}(d) \subseteq \mathit{sel}(d)$*.* $\mathit{dice}_{\mathit{res}}(PCV) = (D_{sel}, \mathit{sel}')$ *with* $\mathit{sel}'(d) = \mathit{res}(d)$ *for* $d \in \mathit{dom}(\mathit{res})$ *and* $\mathit{sel}'(d) = \mathit{sel}(d)$ *for* $d \in D \setminus \mathit{dom}(\mathit{res})$*.*

$\mathit{dice}_{\mathit{res}}(PCV)$ produces a process cube view having the original dimensions. $\mathit{res} \in D_{sel} \not\rightarrow \mathcal{U}_H$ restricts selected dimensions. For example, if $\mathit{res}(\mathit{time}) = \{T_{Jan-2012}, T_{Jan-2013}\}$, $\mathit{res}(\mathit{resource}) = \{\{\mathit{John}\}, \{\mathit{Pete}\}\}$, and $\mathit{res}(\mathit{type}) = \{\{\mathit{gold}, \mathit{silver}\}\}$, then $\mathit{dice}_{\mathit{res}}(PCV)$ restricts the time dimension to two elements (2012 and 2013), the resource dimension to two elements (John and Pete), and the customer type dimension to one element (both gold and silver customers).

7 Roll-Up and Drill-Down

Roll-up and drill-down operations do not remove any events but change the level of granularity of a particular dimension. For example, before drilling down $\mathit{sel}(\mathit{time}) = \{T_{2011}, T_{2012}, T_{2013}\}$ and after drilling down $\mathit{sel}'(\mathit{time}) = \{T_{2011}, T_{Jan-2012}, T_{Feb-2012}, \ldots, T_{Dec-2012}, T_{2013}\}$. Rolling up (sometimes referred to as drilling up) is the reverse. For example, $\mathit{sel}(\mathit{type}) = \{\{\mathit{gold}\}, \{\mathit{silver}\}\}$ is rolled up into $\mathit{sel}'(\mathit{type}) = \{\{\mathit{gold}, \mathit{silver}\}\}$.

Definition 9 (Change Granularity). *Let $PCS = (D, \mathit{type}, \mathit{hier})$ be a process cube structure and $PCV = (D_{sel}, \mathit{sel})$ a view of PCS. Let $d \in D_{sel}$ and $H \in \mathcal{U}_H$ such that:*

- *$H \subseteq \mathit{hier}(d)$,*
- *$\bigcup H = \bigcup \mathit{sel}(d)$, and*
- *for any $V_1, V_2 \in H$: $V_1 \subseteq V_2$ implies $V_1 = V_2$.*

$\mathit{chgr}_{d,H}(PCV) = (D_{sel}, \mathit{sel}')$ with $\mathit{sel}'(d) = H$, and $\mathit{sel}'(d') = \mathit{sel}(d')$ for $d' \in D \setminus \{d\}$.

$\mathit{chgr}_{d,H}(PCV)$ yields a process cube view with the original dimensions D_{sel}. However, dimension d is reorganized in such a way that the result is indeed a view (e.g., elements are not dominating and consistent with the process cube structure) and the set of possible values is unchanged $\bigcup \mathit{sel}'(d) = \bigcup \mathit{sel}(d)$.

8 Conclusion

In this paper, we formalized the notion of process cubes. It gives end users the opportunity to analyze and explore processes interactively on the basis of a multidimensional view on event data. There is no need to extract event logs beforehand like in traditional process mining approaches. Although an initial prototype implementation supporting the main ideas in this paper has been realized [39], many challenges remain. In the remainder, we discuss some of these challenges.

8.1 Comparing and Visualizing Different Cells

First of all, there is the challenge of comparing and visualizing different cells. How to visualize this in an effective manner? Unlike the numerical values shown in traditional OLAP cubes, we need to visualize models that cannot be reduced to simple numbers. Two models many be similar, but their visualizations may be unrelated. This is not just a matter of lay-out. Two process models that are similar from a representational point of view may have very different behaviors and two process models that are different from a representational point of view may have very similar behaviors [1]. Here, we can benefit from research on *configurable process models*. A configurable process model represents a *family* of process models, that is, a model that through configuration can be customized for a particular setting [32, 47, 50, 52]. Process models belonging to such a family need to be related, just like cells in a process cube need to be related to allow for comparison.

Given a process cube, we suggest to visualize the different models with respect to a *cornerstone model*. The different cell models are visualized as edit operations on the cornerstone model. Typical edit operations are: add/remove activity, add/remove edge, hide/insert activity, swap activities, and sequentialize/parallelize activities. These edit operations have costs and are minimized to find the shortest path from the cornerstone model to a particular cell model. Moreover, the edit operations for the different cells are aligned to make the overall understanding of the process cube as simple as possible. One may consider a restricted set of edit operations for block-structured process models [57] to simplify comparison.

There are different approaches to obtain the cornerstone model. The cornerstone model may be selected by the user or may be the first or last model in an array of cells. Moreover, the model may be the Greatest Common Divisor (GCD) or the Least Common Multiple (LCM) of the collection of process models considered [7]. The GCD captures the common parts of the cell models, i.e., all cell models are extensions of the GCD. The LCM embeds all cell models, i.e., all models are restrictions of the LCM. These notions are based on the observation that “hiding” and “blocking” are the essential operators needed for defining inheritance with respect to behavior [8]. The cornerstone model may also be the model closest to all cell models (minimal average edit distance) [38].

8.2 Computing Sublogs and Models Per Cell

Second, there is the problem of performance. The OLAP operations need to be instantaneous to allow for direct user interaction. To realize this, cell results may be pre-computed (materialization of event data and process mining results, e.g., models). However, this may be infeasible in case of many sparse dimensions. Hence, it may be better to do this on-the-fly.

Figure 5 already illustrated the notion of splitting/merging cells horizontally/vertically. We want to do this efficiently for *both* logs and models.

When merging cells one can discover the process model from scratch using the merged event log. As this can be time-consuming, it may be better to merge the process models. Various approaches for merging process models have been proposed in literature [33, 48]. However, these approaches only merge vertically (cf. Figure 5), whereas we also need to support the horizontal merge. Moreover, existing approaches for model merging are not taking into account the event log. Therefore, we would like to develop hybrid approaches that exploit both the existing models and the log to create a merged model that is as close as possible to the original models and the merged event log.

When splitting cells one can discover a process model for each of the smaller event logs. Again this may be time-consuming. Moreover, after splitting, the resulting event logs may be too small to create reliable models. Therefore, we would like to develop hybrid approaches that exploit both the original model and the smaller event logs to create a model for each new cell. For example, the original model may be projected using information from the sublog.

When splitting and merging process cells, one may need to preserve existing relationships between model and event log, e.g., so-called *alignments* [6, 13] need to be split and merged without losing any connections.

8.3 Concept Drift

The time dimension of a process cube has specific properties that can be exploited. For example, the hierarchy of the time dimension can be shared among different applications. Moreover, time introduces particular challenges. For example, processes often change while being analyzed. Therefore, *concept drift* is mentioned as one of the main challenges in the Process Mining Manifesto [36]. Concept drift was been investigated in the context of various data mining problems [62, 37]. In [19] the problem was first investigated in the context of process mining. However, many challenges remain [19, 26], e.g., dealing with incremental drifts and mixtures of periodic drifts.

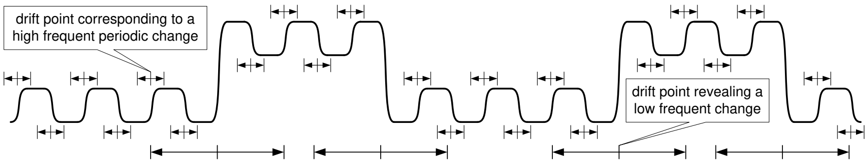
drift point corresponding to a high frequent periodic change
drift point revealing a low frequent change

Fig. 7. A periodically changing processes with two types of drift at different time scales.

Note that the time window dimension in Figure 4 is different from the case type and event class dimensions. In case of short-running cases, we can associate whole cases to time windows. In case of long-running cases, we need to associate individual events to time windows as the process may change while the instance is running. Using carefully selected feature vectors we can analyze drifts using sliding time windows: statistical hypothesis testing will reveal drifts if there are significant differences between two successive windows. A complication is that different types of drifts may be intertwined as illustrated by Figure 7. The drift points are depicted by the bars splitting the double headed arrows: the split arrows represent two consecutive time windows having significant differences. We would also like to relate process changes to contextual elements captured by the cube's dimensions. For example, the time of the day, the weather, the workload, or the type of customer may influence the way cases are handled. As an additional complication, classical conformance notions such as fitness, generalization, and precision [1, 6] cannot be applied to processes that change as one needs to judge the result with respect to a particular time window. Concept drift is also related to *on-the-fly process discovery* [21] where event streams are not stored.

8.4 Distributed Process Mining

Today, there are many types of distributed systems, i.e., systems composed of multiple autonomous computational entities communicating through a network. The terms grid computing, multicore CPU systems, manycore GPU systems, cluster computing, and cloud computing all refer to technologies where different resources are used concurrently to improve performance and scalability. Most data mining techniques can be distributed [23], e.g., there are various techniques for distributed classification, distributed clustering, and distributed association rule mining [17]. These techniques cannot be applied to process mining because events belong to cases and the ordering of events matters. Yet, there is an obvious need for distributed process mining using more efficient and effective discovery techniques. Process mining tasks become challenging when there are hundreds or even thousands of different activities and millions of cases. Typically, process mining algorithms are linear in the number of cases and exponential in the number of different activities.

Process cubes partition event data and therefore may enable *divide-and-conquer approaches* that decompose the event log based on splitting/merging cells horizontally/vertically [3]. This was already illustrated using Figure 5. We are particularly interested in splitting logs horizontally. Thus far we have developed horizontal divide-and-conquer approaches based on SESEs [45, 44], passages [2, 58], and maximal decompositions [5] as a decomposition strategy. As demonstrated in [4, 5] these are merely examples of the broad spectrum of possible techniques to decompose process mining problems. Given the incredible growth of event data, there is an urgent need to explore and investigate the entire spectrum in more detail. Hopefully, such techniques can be used to speed-up OLAP-like operations on process cubes.

Acknowledgements

This work was supported by the Basic Research Program of the National Research University Higher School of Economics (HSE). The author would also like to thank Tatiana Mamaliga for her work on realizing ProCube, a prototype process cube implementation based on ProM and Palo (supervised by the author and Joos Buijs).

References

1. W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer-Verlag, Berlin, 2011.
2. W.M.P. van der Aalst. Decomposing Process Mining Problems Using Passages. In S. Haddad and L. Pomello, editors, *Applications and Theory of Petri Nets 2012*, volume 7347 of *Lecture Notes in Computer Science*, pages 72–91. Springer-Verlag, Berlin, 2012.
3. W.M.P. van der Aalst. Distributed Process Discovery and Conformance Checking. In J. de Lara and A. Zisman, editors, *International Conference on Fundamental Approaches to Software Engineering (FASE 2012)*, volume 7212 of *Lecture Notes in Computer Science*, pages 1–25. Springer-Verlag, Berlin, 2012.
4. W.M.P. van der Aalst. A General Divide and Conquer Approach for Process Mining. In M. Ganzha, L. Maciaszek, and M. Paprzycki, editors, *Federated Conference on Computer Science and Information Systems (FedCSIS 2013)*, pages 1–10. IEEE Computer Society, 2013.
5. W.M.P. van der Aalst. Decomposing Petri Nets for Process Mining: A Generic Approach. *Distributed and Parallel Databases*, 31(4):471–507, 2013.
6. W.M.P. van der Aalst, A. Adriansyah, and B. van Dongen. Replaying History on Process Models for Conformance Checking and Performance Analysis. *WIREs Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.
7. W.M.P. van der Aalst and T. Basten. Identifying Commonalities and Differences in Object Life Cycles using Behavioral Inheritance. In J.M. Colom and M. Koutny, editors, *Application and Theory of Petri Nets 2001*, volume 2075 of *Lecture Notes in Computer Science*, pages 32–52. Springer-Verlag, Berlin, 2001.
8. W.M.P. van der Aalst and T. Basten. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theoretical Computer Science*, 270(1-2):125–203, 2002.
9. W.M.P. van der Aalst and S. Dustdar. Process Mining Put into Context. *IEEE Internet Computing*, 16(1):82–86, 2012.
10. W.M.P. van der Aalst, H.A. Reijers, and M. Song. Discovering Social Networks from Event Logs. *Computer Supported Cooperative work*, 14(6):549–593, 2005.
11. W.M.P. van der Aalst, V. Rubin, H.M.W. Verbeek, B.F. van Dongen, E. Kindler, and C.W. Günther. Process Mining: A Two-Step Approach to Balance Between Underfitting and Overfitting. *Software and Systems Modeling*, 9(1):87–111, 2010.
12. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
13. A. Adriansyah, B. van Dongen, and W.M.P. van der Aalst. Conformance Checking using Cost-Based Fitness Analysis. In C.H. Chi and P. Johnson, editors, *IEEE International Enterprise Computing Conference (EDOC 2011)*, pages 55–64. IEEE Computer Society, 2011.
14. A. Adriansyah, B.F. van Dongen, and W.M.P. van der Aalst. Towards Robust Conformance Checking. In M. zur Muehlen and J. Su, editors, *BPM 2010 Workshops,*

Proceedings of the Sixth Workshop on Business Process Intelligence (BPI2010), volume 66 of *Lecture Notes in Business Information Processing*, pages 122–133. Springer-Verlag, Berlin, 2011.

15. A. Adriansyah, N. Sidorova, and B.F. van Dongen. Cost-based Fitness in Conformance Checking. In *International Conference on Application of Concurrency to System Design (ACSD 2011)*, pages 57–66. IEEE Computer Society, 2011.
16. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, volume 1377 of *Lecture Notes in Computer Science*, pages 469–483. Springer-Verlag, Berlin, 1998.
17. R. Agrawal and J.C. Shafer. Parallel Mining of Association Rules. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):962–969, 1996.
18. R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Process Mining Based on Regions of Languages. In G. Alonso, P. Dadam, and M. Rosemann, editors, *International Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 375–383. Springer-Verlag, Berlin, 2007.
19. R.P. Jagadeesh Chandra Bose, W.M.P. van der Aalst, I. Zliobaite, and M. Pechenizkiy. Handling Concept Drift in Process Mining. In H. Mouratidis and C. Rolland, editors, *International Conference on Advanced Information Systems Engineering (Caise 2011)*, volume 6741 of *Lecture Notes in Computer Science*, pages 391–405. Springer-Verlag, Berlin, 2011.
20. J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst. Towards Cross-Organizational Process Mining in Collections of Process Models and their Executions. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *Business Process Management Workshops, International Workshop on Process Model Collections (PMC 2011)*, volume 100 of *Lecture Notes in Business Information Processing*, pages 2–13. Springer-Verlag, Berlin, 2012.
21. A. Burattin, A.Sperduti, and W.M.P. van der Aalst. Heuristics Miners for Streaming Event Data. *CoRR*, abs/1212.6383, 2012.
22. T. Calders, C. Guenther, M. Pechenizkiy, and A. Rozinat. Using Minimum Description Length for Process Mining. In *ACM Symposium on Applied Computing (SAC 2009)*, pages 1451–1455. ACM Press, 2009.
23. M. Cannataro, A. Congiusta, A. Pugliese, D. Talia, and P. Trunfio. Distributed Data Mining on Grids: Services, Tools, and Applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(6):2451–2465, 2004.
24. J. Carmona and J. Cortadella. Process Mining Meets Abstract Interpretation. In J.L. Balcazar, editor, *ECML/PKDD 210*, volume 6321 of *Lecture Notes in Artificial Intelligence*, pages 184–199. Springer-Verlag, Berlin, 2010.
25. J. Carmona, J. Cortadella, and M. Kishinevsky. A Region-Based Algorithm for Discovering Petri Nets from Event Logs. In *Business Process Management (BPM2008)*, pages 358–373, 2008.
26. J. Carmona and R. Gavalda. Online techniques for dealing with concept drift in process mining. In *Advances in Intelligent Data Analysis XI*, volume 172, pages 90–102. Springer-Verlag, Berlin, 2012.
27. S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technology. *ACM Sigmod Record*, 26(1):65–74, 1997.
28. J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.

29. J.E. Cook and A.L. Wolf. Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. *ACM Transactions on Software Engineering and Methodology*, 8(2):147–176, 1999.
30. W. Gaaloul, K. Gaaloul, S. Bhiri, A. Haller, and M. Hauswirth. Log-Based Transactional Workflow Mining. *Distributed and Parallel Databases*, 25(3):193–240, 2009.
31. S. Goedertier, D. Martens, J. Vanthienen, and B. Baesens. Robust Process Discovery with Artificial Negative Events. *Journal of Machine Learning Research*, 10:1305–1340, 2009.
32. F. Gottschalk, W.M.P. van der Aalst, M.H Jansen-Vullers, and M. La Rosa. Configurable Workflow Models. *International Journal of Cooperative Information Systems*, 17(2):177–221, 2008.
33. F. Gottschalk, T. Wagemakers, M.H. Jansen-Vullers, W.M.P. van der Aalst, and M. La Rosa. Configurable Process Models: Experiences From a Municipality Case Study. In P. van Eck, J. Gordijn, and R. Wieringa, editors, *Advanced Information Systems Engineering, Proceedings of the 21st International Conference on Advanced Information Systems Engineering (CAiSE'09)*, volume 5565 of *Lecture Notes in Computer Science*, pages 486–500. Springer-Verlag, Berlin, 2009.
34. C.W. Günther and W.M.P. van der Aalst. Fuzzy Mining: Adaptive Process Simplification Based on Multi-perspective Metrics. In G. Alonso, P. Dadam, and M. Rosemann, editors, *International Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 328–343. Springer-Verlag, Berlin, 2007.
35. M. Hilbert and P. Lopez. The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, 332(6025):60–65, 2011.
36. IEEE Task Force on Process Mining. Process Mining Manifesto. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *Business Process Management Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 169–194. Springer-Verlag, Berlin, 2012.
37. M. van Leeuwen and A. Siebes. StreamKrimp: Detecting Change in Data Streams. In *Machine Learning and Knowledge Discovery in Databases*, volume 5211 of *Lecture Notes in Computer Science*, pages 672–687. Springer-Verlag, Berlin, 2008.
38. C. Li, M. Reichert, and A. Wombacher. The MINADEPT Clustering Approach for Discovering Reference Process Models Out of Process Variants. *International Journal of Cooperative Information Systems*, 19(3-4):159–203, 2010.
39. T. Mamaliga. Realizing a Process Cube Allowing for the Comparison of Event Data. Master's thesis, Eindhoven University of Technology, Eindhoven, 2013.
40. J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. Byers. Big Data: The Next Frontier for Innovation, Competition, and Productivity. McKinsey Global Institute, 2011.
41. A.K. Alves de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. Genetic Process Mining: An Experimental Evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, 2007.
42. J. Munoz-Gama and J. Carmona. A Fresh Look at Precision in Process Conformance. In R. Hull, J. Mendling, and S. Tai, editors, *Business Process Management (BPM 2010)*, volume 6336 of *Lecture Notes in Computer Science*, pages 211–226. Springer-Verlag, Berlin, 2010.
43. J. Munoz-Gama and J. Carmona. Enhancing Precision in Process Conformance: Stability, Confidence and Severity. In N. Chawla, I. King, and A. Sperduti, editors, *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*, pages 184–191, Paris, France, April 2011. IEEE.

44. J. Munoz-Gama, J. Carmona, and W.M.P. van der Aalst. Conformance Checking in the Large: Partitioning and Topology. In F. Daniel, J. Wang, and B. Weber, editors, *International Conference on Business Process Management (BPM 2013)*, volume 8094 of *Lecture Notes in Computer Science*, pages 130–145. Springer-Verlag, Berlin, 2013.
45. J. Munoz-Gama, J. Carmona, and W.M.P. van der Aalst. Hierarchical Conformance Checking of Process Models Based on Event Logs. In J.M. Colom and J. Desel, editors, *Applications and Theory of Petri Nets 2013*, volume 7927 of *Lecture Notes in Computer Science*, pages 291–310. Springer-Verlag, Berlin, 2013.
46. J.T.S. Ribeiro and A.J.M.M. Weijters. Event Cube: Another Perspective on Business Processes. In *OTM 2011*, volume 7044 of *Lecture Notes in Computer Science*, pages 274–283. Springer-Verlag, Berlin, 2011.
47. M. La Rosa, M. Dumas, A. ter Hofstede, and J. Mendling. Configurable Multi-Perspective Business Process Models. *Information Systems*, 36(2):313–340, 2011.
48. M. La Rosa, M. Dumas, R. Uba, and R.M. Dijkman. Business Process Model Merging: An Approach to Business Process Consolidation. *ACM Transactions on Software Engineering and Methodology*, 22(2), 2012.
49. M. La Rosa, H.A. Reijers, W.M.P. van der Aalst, R.M. Dijkman, J. Mendling, M. Dumas, and L. Garcia-Banuelos. APROMORE: An Advanced Process Model Repository. *Expert Systems With Applications*, 38(6):7029–7040, 2011.
50. M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32(1):1–23, 2007.
51. A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1):64–95, 2008.
52. A. Schnieders and F. Puhlmann. Variability Mechanisms in E-Business Process Families. In W. Abramowicz and H.C. Mayr, editors, *Proceedings of the 9th International Conference on Business Information Systems (BIS'06)*, volume 85 of *LNI*, pages 583–601. GI, 2006.
53. A. Sheth. A New Landscape for Distributed and Parallel Data Management. *Distributed and Parallel Databases*, 30(2):101–103, 2012.
54. M. Sole and J. Carmona. Process Mining from a Basis of Regions. In J. Lilius and W. Penczek, editors, *Applications and Theory of Petri Nets 2010*, volume 6128 of *Lecture Notes in Computer Science*, pages 226–245. Springer-Verlag, Berlin, 2010.
55. M. Song and W.M.P. van der Aalst. Supporting Process Mining by Showing Events at a Glance. In K. Chari and A. Kumar, editors, *Proceedings of 17th Annual Workshop on Information Technologies and Systems (WITS 2007)*, pages 139–145, Montreal, Canada, December 2007.
56. M. Song and W.M.P. van der Aalst. Towards Comprehensive Support for Organizational Mining. *Decision Support Systems*, 46(1):300–317, 2008.
57. J. Vanhatalo, H. Völzer, and J. Koehler. The Refined Process Structure Tree. *Data and Knowledge Engineering*, 68(9):793–818, 2009.
58. H.M.W. Verbeek and W.M.P. van der Aalst. Decomposing Replay Problems: A Case Study. BPM Center Report BPM-13-09, BPMcenter.org, 2013.
59. J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens. A Robust F-measure for Evaluating Discovered Process Models. In N. Chawla, I. King, and A. Sperduti, editors, *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*, pages 148–155, Paris, France, April 2011. IEEE.
60. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.

61. J.M.E.M. van der Werf, B.F. van Dongen, C.A.J. Hurkens, and A. Serebrenik. Process Discovery using Integer Linear Programming. *Fundamenta Informaticae*, 94:387–412, 2010.
62. G. Widmer and M. Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning*, 23:69–101, 1996.