Multidimensional Process Mining using Process Cubes

Alfredo Bolt and Wil M.P. van der Aalst

Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands
{a.bolt,w.m.p.v.d.aalst}@tue.nl

Abstract. Process mining techniques enable the analysis of processes using event data. For structured processes without too many variations, it is possible to show a relative simple model and project performance and conformance information on it. However, if there are multiple classes of cases exhibiting markedly different behaviors, then the overall process will be too complex to interpret. Moreover, it will be impossible to see differences in performance and conformance for the different process variants. The different process variations should be analysed separately and compared to each other from different perspectives to obtain meaningful insights about the different behaviors embedded in the process. This paper formalizes the notion of *process cubes* where the event data is presented and organized using different dimensions. Each cell in the cube corresponds to a set of events which can be used as an input by any process mining technique. This notion is related to the well-known OLAP (Online Analytical Processing) data cubes, adapting the OLAP paradigm to event data through *multidimensional process mining*. This adaptation is far from trivial given the nature of event data which cannot be easily summarized or aggregated, conflicting with classical OLAP assumptions. For example, multidimensional process mining can be used to analyze the different versions of a sales processes, where each version can be defined according to different dimensions such as location or time, and then the different results can be compared. This new way of looking at processes may provide valuable insights for process optimization.

Keywords: Process Cube, Process Mining, OLAP, Comparative Process Mining

1 Introduction

Process Mining can be seen as the missing link between model-based process analysis (e.g., simulation and verification) and data-oriented analysis techniques such as machine learning and data mining [1]. It seeks the "confrontation" between real event data and process models (automatically discovered or hand-made). Classical process mining techniques focus on analysing a process as a whole, but in this paper we focus on isolating different *process behaviors* (versions) and present them in a way that facilitates their *comparison* by approaching process mining in a multidimensional perspective.

Multidimensional process mining has been approached recently by some authors. The *event cube* approach described in [2] presents an exploratory view on the applications of OLAP operations using *events*. The *process cube* approach is introduced by the second author in [3] with an initial prototype implementation [4]. The process cube notion was proven useful in case studies [5, 6]. These approaches have established a *conceptual* framework for process cubes, however, they still present some conceptual limitations. One of the limitations of [3] is related to concurrency issues (e.g. derived properties are created on the *event base* which may be used with many *process cube structures*, which would force all the dimensions that correspond to a specific property to have exactly the same meaning and value set. This is an undesired behavior when for example, calculating in different *process cube structures* a dimension *customer type* according to different criteria). Other limitations are the structure within dimensions (e.g. there is no composition of attributes and no hierarchies of aggregation, therefore no *roll-up* and *drill-down* directions) and the (lack of) granularity-level definitions (used for defining the cube cells distribution and filter the events in each cell). In this paper we provide an improved formalization of the *process cube* conceptual framework.

The idea is related to the well-known OLAP multidimensional paradigm [7]. OLAP techniques organize the data under multiple combinations of dimensions and typically numerical measures, and accessing the data through different OLAP operations such as slicing, dicing, rolling up and drilling down. Lots of research have been conducted to deal with OLAP technical issues such as the materialization process. An extensive overview of such approaches can be found in [8]. The application of OLAP on non-numerical data is increasingly being explored. Temporal series, graphs and complex event sequences are possible applications [9–11]. However, there are two significant differences between OLAP and Process Cubes: Summarizability and Representation. The first refers to the classic OLAP cubes assumption on the summarizability of facts. This allows for precomputations of the different multidimensional perspectives of the cube, which provides real-time (On-Line) analysis capabilities. Some authors have studied summarizability issues in OLAP [15, 16] and attempt to solve it by introducing rules and constraints to the data model. In Process Cubes, summarizability is not guaranteed because of the process-oriented nature of the event data used. In Process Mining, each event is related to one or more traces, and the relevance of an event as data is given mostly by its relations with other events within those traces. One cannot simply merge or split Process Cube cells as summarizable OLAP cells because events are ordered, and any slight change in that ordering may change the whole representation of the cell where that event is being contained. The second refers to classical OLAP relying on the aggregation of facts for reducing a set of values into a single value that can be represented in many ways. On the other hand, Process Cubes have to deal with a much more complex representation of data. Process Cube cells are associated to process models and not just event data, and both are directly related. Observed and modeled behavior can be compared, process models can be discovered from events, and events can be used to replay behavior into otherwise static process models.

The remainder is organized as follows. In Sec 2. we define the process cube notion as a means for viewing event data from different perspectives. Sec 3. presents our implementation of process cubes. In Sec 4. we discuss the experiments and benefits that can be achieved through our approach. Finally Sec 5. concludes the paper by discussing some challenges and future work.

2 Process Cubes

In this section we will formalize the notion of a process cube, defining all of its inner components. A process cube is formed by a structure that describes the "shape" of the cube (distribution of cells) and by the real data that will be used as a basis to "fill" those cells.

2.1 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 machine) executing or initiating the activity, the *timestamp* of the event, or *data elements* recorded with the event.

An *event collection* is a set of events that have certain properties, but no defined *cases* and *activities*. Table 1 shows a small fragment of some larger event collection. Each event has a unique id and several properties. For example, event 0001 is an instance of action A that occurred on December 28th of 2014 at 6:30 am, was executed by John, and costed 100 euros. An event collection can be transformed into an event log by selecting event properties (or attributes) as *case_id* and *activity_id*. For example, in Table 1, *sales order* could be the *case_id* and *action* could be the *activity_id* of an event log containing all events of the event collection.

Table 1. A fragment of an event collection: each row corresponds to an event.

event_id	sales order	timestamp	action	resource	cost
0001	1	28-12-2014:06.30	A	John	100
0002	1	28-12-2014:07.15	B	Anna	
0003	1	28-12-2014:08.45	C	John	
0004	2	28-12-2014:12.20	A	Peter	150
0005	1	28-12-2014:20.28	D	Mike	
0006	2	28-12-2014:23.30	C	Anna	
...	...	...	...	...	...

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. The events in the event base are used to populate the cells in the cube. Throughout the paper we assume the following universes.

Definition 1 (Universes) *$\mathcal{U}_V$ is the universe of all attribute values (e.g., strings, numbers, etc..). $\mathcal{U}_S = \mathcal{P}(\mathcal{U}_V)$ is the universe of value sets. $\mathcal{U}_A$ is the universe of all attribute names (e.g., year, action, etc...).*

Note that $v \in \mathcal{U}_V$ is a single value (e.g., $v = 5$), $V \in \mathcal{U}_S$ is a set of values (e.g., $V = \{Europe,\ America\}$), $a \in \mathcal{U}_A$ is a single attribute name (e.g., *age*).

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 $\pi \in P \to (E \not\rightarrow \mathcal{U}_V)$. For any property $p \in P$, $\pi(p)$ (denoted π_p) is a partial function mapping events into 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 a property p and we write $\pi_p(e) = \perp$ to indicate this.*

An event base is created from an event collection like the one presented in Table 1. If we transform this table into an EB, then the set of events E consist of all different elements of the *event_id* column of Table 1. In this case, $E = \{0001,0002,0003,0004,0005,0006,...\}$. The set of properties P is the set of column headers of Table 1, with the exception of *event_id*. In this case, $P = \{sales\ order,\ timestamp,\ action,\ resource,\ cost\}$. The function π retrieves the value of each row (event) and column (property) combination (cell) in Table 1. For example, the value of the property *action* for the event 0001 is given by $\pi_{action}(0001) = A$. In the case that this value is empty in the table, we will use $\perp$ to denote it in the EB (e.g., $\pi_{cost}(0002) = \perp$).

Note that an event identifier (*event_id*) $e \in E$ does not have a meaning, but it is unique for each event.

2.2 Process Cube Structure

Independent of the event base EB we define the *structure* of the process cube. A *Process Cube Structure (PCS)* is fully characterized by the set of *dimensions* defined for it, each dimension having its own *hierarchy*.

Before defining the concepts of hierarchy and dimension, we need to define some basic graph properties.

Definition 3 (Directed Acyclic Graph) A *directed acyclic graph (DAG) is a pair $G = (N, E)$ where N is a set of nodes and $E \subseteq N \times N$ a set of edges connecting these nodes, where:*

– $n_1, n_2 \in N, n_1 \neq n_2 : e_1 = (n_1, n_2) \in E$ is a *directed edge* that starts in n_1 and ends in n_2,
– *A walk $W \in E^*$ with a length of $|W| \geq 1$ is an ordered list of directed edges $W = (e_1, ..., e_k)$ with $e_j \in E : e_j = (n_j, n_{j+1}) \Rightarrow e_{j+1} = (n_{j+1}, n_{j+2}),\ 1 \leq j < k \in \mathbb{N}$, and*
– $\forall n \in N$: there is no walk $W \in E^*$ that starts and ends in n.

Note that there cannot be any directed cycles of any length in a DAG. For example, part (1) in Fig 1 shows a DAG with nodes: $\{$*City,Country,etc...*$\}$.

Definition 4 (Dimension) *A dimension is a pair $d = ((A, H), valueset)$ where the* hierarchy *(A, H) is a DAG with nodes $A \subseteq \mathcal{U}_A$ (attributes) and a set of directed edges $H \subseteq A \times A$, and $valueset \in A \rightarrow \mathcal{U}_S$ is a function defining the possible set of values for each attribute.*

The attributes in A are unique. The set of directed edges H defines the navigation directions for exploring the dimension. An edge $(a_1, a_2) \in H$ means that attribute a_1 can be *rolled up* to attribute a_2 (defined in Sec 2.5). A dimension should describe events from a single perspective through any combination of its attributes (e.g., attributes *city* and *country* can describe a *Location*) where attributes describe the perspective from higher or lower levels of detail (e.g., *city* describes a *Location* in a more fine-grained level than *country*). However, this is not strict and users can define dimensions as they want.

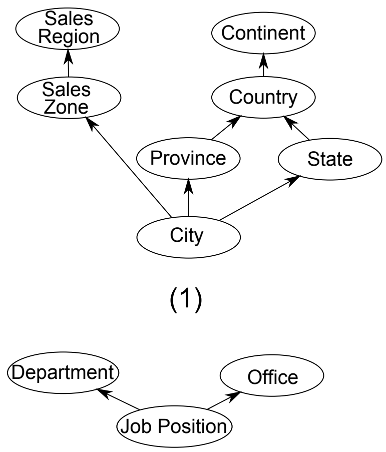
An attribute $a \in A$ has a $valueset(a)$ that is the set of possible values and typically only a subset of those values are present in a concrete instance of the process cube. For example, $valueset(age) = \{1, 2, ..., 120\}$ for $age \in A$. Another example: $valueset(cost) = \mathbb{N}$ allows for infinitely many possible values. We introduce the notation A_d to refer to the set of attributes A of the dimension d, and $\mathcal{U}_D$ as the universe of all possible dimensions. Fig 1. shows some examples of dimensions, each containing a DAG and a valueset function.

Definition 5 (Process Cube Structure) *A process cube structure is a set of dimensions $PCS \subseteq \mathcal{U}_D$, where for any two dimensions $d_1, d_2 \in PCS, d_1 \neq d_2 : A_{d_1} \cap A_{d_2} = \emptyset$.*

All dimensions in a process cube structure are independent from each other, this means that they do not have any attributes in common, so all attributes are unique, however, their value sets might have common values. We introduce the notation A_{pcs} to refer to the union of all sets of attributes $\bigcup_{d \in PCS} A_d$.

2.3 Compatibility

A process cube structure PCS and an event base EB are independent elements, where the PCS is the structure and the EB is the content of the cube. To make sure that we can use them together, we need to relate them through a mapping function and then check whether they are compatible.



(2)

Dimension: Location

Attribute (a)	Valueset(a)
Sales Region	{North Europe, East Asia, South America, etc.}
Sales Zone	{Iberian Penunsula, South Germany, etc.}
Continent	{Iberian Penunsula, South Germany, etc.}
Country	{Netherlands, Chile, Spain, etc.}
Province	{Noord Brabant, Andalucia, Siberia, etc.}
State	{California, New York, etc.}
City	{Eindhoven, Amsterdam, Santiago, Madrid, etc.}

Dimension: Organigram

Attribute (a)	Valueset(a)
Office	{Boston, London, Eindhoven, etc.}
Department	{Marketing, Operations, National Sales, International Sales, etc.}
Job Position	{Software Engineer, Sales Executive, etc.}

Fig. 1. Example of two dimensions (Location, Organigram), both conformed by a directed acyclic graph (1) and a valueset function (2).

Definition 6 (Mapper) *A mapper is a triplet $M = (PCS, EB, R)$ where PCS is a process cube structure, $EB = (E, P, \pi)$ is an event base and R is a function: $R \in A_{pcs} \rightarrow (\mathcal{P}(P) \times (E \nrightarrow \mathcal{U}_V))$. For an attribute $a \in A_{pcs}$, $R(a) = (P', g_a)$ where $P' = \{p_1, ..., p_n\} \in \mathcal{P}(P)$ is a set of properties and g_a is a calculation function mapping events into values used to calculate the value of a, so that for any event $e \in E$ with $\forall p \in P' : e \in dom(\pi_p)$, the value of attribute a for the event e is given by $g_a(e) = f(\pi_{p_1}(e), ..., \pi_{p_n}(e)) = v \in \mathcal{U}_V$. If for any $p \in P' : e \notin dom(\pi_p)$ then event e does not have the property p and the value of the attribute a cannot be calculated, so we write $g_a(e) =\perp$ to indicate this.*

Note that each attribute is related to one set of properties which is used to calculate the value of the attribute for any event through a specific calculation function. For example, an attribute *day type* = {*weekend,weekday*} can be calculated using the event properties {*day,month,year*} according to some specific calendar rules. Another example is an attribute *age* which can be calculated from properties {*birthday,timestamp*} by the function: $g_{age}(e) = \pi_{timestamp}(e) - \pi_{birthday}(e)$.

A set of properties can be used by more than one attributes producing different results if the calculation function is different. For example, in sales one could use the set of properties {*purchase amount, purchase num*} to classify customers into an attribute *customer type* = {*Gold, Silver*} (i.e., if *purchase amount* > 50 and *purchase num >10*, then *customer type = Silver*) and at the same time to detect fraud into an attribute *fraud risk* = {*High,Low*} (i.e., if *purchase amount* > 100000 and *purchase num* = 1, then *fraud risk = High*).

Given a mapper $M = (PCS, EB, R)$ we say that PCS and EB are *compatible* through R, making all views of PCS also compatible with the EB.

2.4 Process Cube View

Once a proces cube structure is defined, it does not change. While applying typical OLAP operations such as slice, dice, roll up and drill down (defined in Sec 2.5) we only change the way we are visualizing the cube and its content. A *process cube view* defines the visible part of the process cube structure.

Definition 7 (Process Cube View) *Let PCS be a process cube structure. A process cube view is a triplet $PCV = (D_{vis}, sel, gran)$ such that:*

- *$D_{vis} \subseteq PCS$ are the visible dimensions,*
- *$sel \in A_{pcs} \rightarrow \mathcal{U}_S$ is a function selecting a part of the value set of the attributes of each dimension, such that for any $a \in A_{pcs} : sel(a) \subseteq valueset(a)$, and*
- *$gran \in D_{vis} \rightarrow \mathcal{U}_A$ is a function defining the granularity for each one of the visible dimensions.*

The *sel* function selects sets of values per attribute (including attributes in not visible dimensions). For example, in the dimension *Organigram* in Fig 1, one could select the job position *Sales Executive*, but many departments could have that same job position, so we could also select the department *National Sales* to only see the Sales Executives that work in National Sales. On the other hand, if this selection is done incorrectly, it might lead to empty results. For example in the dimension *Location* in Fig 1. one could select the city *Eindhoven* and the country *Spain* and this would produce empty results since no event can have both values. In our approach we made this as flexible as possible, so it is up to the user to check if the selection is done properly.

For each visible dimension, the *gran* function defines one of its attributes as the granularity. This will be used to define the *cell set* of the cube where each value of the granularity attribute corresponds to a cell. For example, in the dimension *Organigram* in Fig 1, one could define the Job Title as granularity.

Many different process cube views can be obtained form the same process cube structure. For example, Fig. 2. shows two process cube views obtained from the same process cube structure.

Definition 8 (Cell Set) *Let PCS be a process cube structure and $PCV = (D_{vis}, sel, gran)$ be a view over PCS with $D_{vis} = \{d_1, ..., d_n\}$. The* cell set *of PCV is defined as $CS_{pcv} = AV_{d_1} \times ... \times AV_{d_n}$, where for any $d_i \in D_{vis} : AV_{d_i} = gran(d_i) \times sel(gran(d_i))$ is a set of attribute-value sets.*

Although the term *cube* suggests a three dimensional object a process cube can have any number of visible dimensions.

A cell set CS is the set of visible cells of the process cube view. For example, for a process cube view with visible dimensions *Location* and $Time$ with their granularity set to: $gran(Location) = \{City\}$ and $gran(Time) = \{Year\}$ and the selected values of those attributes were: *sel(City)* = {*Eindhoven, Amsterdam*} and *sel(Year)* = {*2013,2014*}, the cube would have the following 4 cells: {(*City, Eindhoven*), (*Year, 2013*)}, {(*City, Eindhoven*), (*Year, 2014*)}, {(*City, Amsterdam*), (*Year, 2013*)}, and {(*City, Amsterdam*), (*Year, 2014*)}.

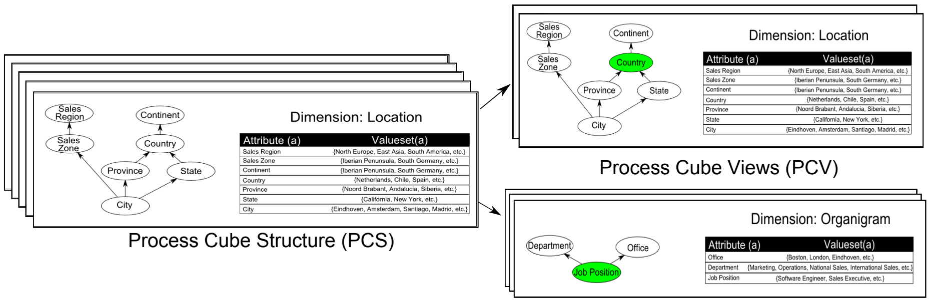


Fig. 2. Example of two PCVs created from the same PCS, both selecting some dimensions, selecting a part of the valuesets, and selecting attributes as granularity for the selected dimensions.

2.5 Process Cube Operations

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

The *slice operation* produces a new cube by allowing the analyst to filter (pick) specific values for attributes within one of the dimensions, while removing that dimension from the visible part of the cube.

The *dice operation* produces a subcube by allowing the analyst to filter (pick) specific values for one of the dimensions. No dimensions are removed in this case, but only the selected values are considered. Fig 3. illustrates the notions of slicing and dicing. For both operations, the same filtering was applied. In the case of the slice operation, the *Location* dimension is no longer visible, but in dice one could still use that dimension for further operations (i.e., drilling down to *City*) keeping the same dimensions visible.

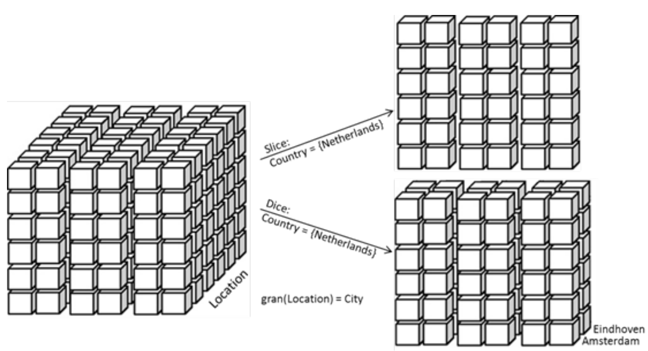


Fig. 3. Slice and Dice operations

The *roll up* and *drill down* operations do not remove any dimensions or filter any values, but only change the level of granularity of a specific dimension. Fig 4. shows the concept of drilling down and rolling up. These operations are intended to show the same data with more or less detail (granularity). However, this is not guaranteed as it depends on the dimension definition.

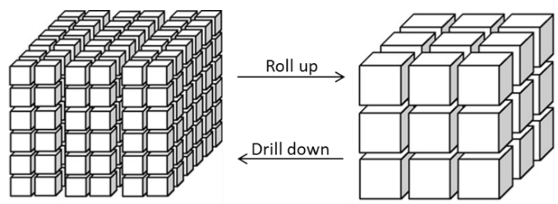


Fig. 4. Roll up and Drill down operations

Definition 9 (Slice) *Let PCS be a process cube structure and let $PCV = (D_{vis}, sel, gran)$ be a view of PCS. We define* slice *for a dimension $d \in D_{vis}$: $d = ((A, H), valueset)$ and a* filtering *function $fil \in A_d \to \mathcal{U}_S$ where for any $a \in A_d : fil(a) \subseteq valueset(a)$, as: $slice_{d,fil}(PCV) = (D'_{vis}, sel', gran)$, where:*

- *$D'_{vis} = D_{vis} \setminus \{d\}$ is the new set of visible dimensions, and*
- *$sel' \in A_{pcs} \to \mathcal{U}_S$ is the new selection function, where:*
  - *for any $a \in A_d : sel'(a) = fil(a)$, and*
  - *for any $a \in A_{pcs} \setminus A_d : sel'(a) = sel(a)$.*

The *slice* operation produces a new process cube view. Note that d is no longer a visible dimension: $d \notin D'_{vis}$ but it will be used for filtering events. The new sel' function will still be valid as a value set selection function when filtering. Also, note that the *gran* function remains unaffected. For example, for sales data one could slice the cube for a dimension *Location* for City *Eindhoven*, the Location dimension is removed from the cube and only sales of the stores in Eindhoven are considered. One could also do more complex slicing. For example, for a dimension *time*, one could slice that dimension and select years 2013 and 2014 and months *January* and *February*, then the time dimension is removed from the cube and only sales in January or February of years 2013 or 2014 are considered.

Definition 10 (Dice) *Let PCS be a process cube structure and let $PCV = (D_{vis}, sel, gran)$ be a view of PCS. We define* dice *for a dimension $d \in D_{vis}$: $d = ((A, H), valueset)$ and a* filtering *function $fil \in A_d \to \mathcal{U}_S$ where for any $a \in A_d : fil(a) \subseteq valueset(a)$, as: $slice_{d,fil}(PCV) = (D_{vis}, sel', gran)$, where:*

- *$sel' \in A_{pcs} \rightarrow \mathcal{U}_S$ is the new selection function, where:*
  - *for any $a \in A_d : sel'(a) = fil(a)$, and*
  - *for any $a \in A_{pcs} \setminus A_d : sel'(a) = sel(a)$.*

The *dice* operation is very similar to the *slice* operation defined previously, where the only difference is that in *dice* the dimension is not removed from D_{vis}.

Definition 11 (Change Granularity) *Let PCS be a process cube structure and $PCV = (D_{vis}, sel, gran)$ a view of PCS. We define* chgr *for a dimension $d \in D_{vis} : d = ((A, H), valueset)$ and an attribute $a \in A_d$ as: $chgr_{d,a}(PCV) = (D_{vis}, sel, gran')$, where $gran'(d) = a$, and for any $d' \in D_{vis} \setminus \{d\} : gran'(d') = gran(d')$.*

This operation produces a new process cube view and allows us to set any attribute of the dimension d as the new granularity for that dimension, leaving any other dimension untouched. However, typical OLAP cubes allow the user to "navigate" through the cube using roll up and drill down operations, changing the granularity in a guided way through the hierarchy of the dimension. Note that D_{vis} and *sel* always remain unaffected when changing granularity. Now we define the roll up and drill down operation using the previously defined *chgr* function.

Definition 12 (Roll up and Drill Down) *Let PCS be a process cube structure, $PCV = (D_{vis}, sel, gran)$ a view of PCS with a dimension $d \in D_{vis} : d = ((A, H), valueset)$. We can* roll up *the dimension d if $\exists a \in A_d : (gran(d), a) \in H$. The result is a more coarse-grained cube: $rollup_{d,a}(PCV) = chgr_{d,a}(PCV)$. We can* drill down *the dimension d if $\exists a \in A_d : (a, gran(d)) \in H$. The result is a more fine-grained cube: $drilldown_{d,a}(PCV) = chgr_{d,a}(PCV)$.*

If there is more than one attribute a that the dimension could be *rolled up* or *drilled down* to, then any of those attributes can be a valid target, but we can pick only one each time. For example, for a dimension *Location* described in Fig 1, we could roll up the dimension from *City* to *Province, State* or *Sales Zone*.

2.6 Materialized Process Cube View

Once we selected a part of the cube structure, and there is a cell set defined as the visible part of the cube, now we have to add content to those cells. In other words, we have to add *events* to these cells so they can be used by process mining algorithms.

Definition 13 (Materialized Process Cube View) *Let $M = (PCS, EB, R)$ be a mapper with PCS being a process cube structure, $EB = (E, P, \pi)$ being an event base, and let $PCV = (D_{vis}, sel, gran)$ be a view over PCS with a cell* *set* CS_{pcv}*. The* materialized process cube view *for* PCV *and* EB *is defined as a function* $MPCV_{pcv,eb} \in CS_{pcv} \rightarrow \mathcal{P}(E)$ *that relates sets of events to cells* $c \in CS_{pcv} : MPCV_{pcv,eb}(c) = \{e \in E | (1) \wedge (2)\}$*, where:*

(1) $\forall d \in D_{vis}, R(gran(d)) = (P', g_{gran(d)}) : (gran(d), g_{gran(d)}(e)) \in c$
(2) $\forall a \in A_{pcs}, R(a) = (P', g_a) : g_a(e) \in sel(a)$

The first condition (1) relates an event to a cell if the event property values are related to the (attribute,value) pairs that define that cell. For example, for a cell $c = \{(\textit{year,2012}),(\textit{city,Eindhoven})\}$ one could relate all events that have both attribute values to that cell.

The second condition (2) is to check if the events related to each cell are not filtered out by any other attribute of any dimension of the process cube structure. Note that this condition becomes specially useful when slicing dimensions.

Fig 5 shows an example of a materialized process cube view. Each of the selected dimensions conform the cell distribution of the cube, and the events in the event base are mapped to these cells.

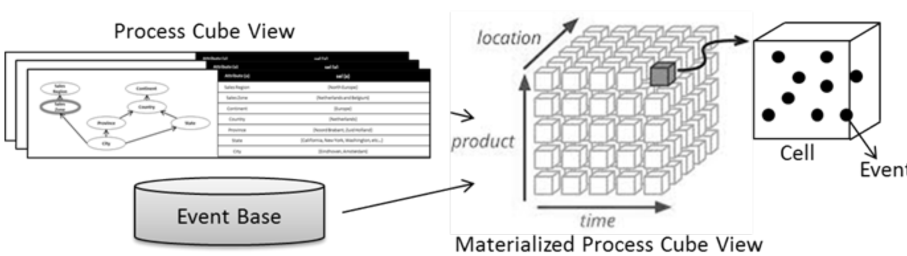


Fig. 5. Example of a materialized process cube view (MPCV) for an event base (EB) and a process cube view (PCV). The cells of MPCV contain events.

Normally events are related to specific activities or facts that happen in a process, and they are grouped in cases. In order to transform the set of events of a cell into an event log, we must define a *case id* to identify cases and an *activity id* to identify activities. The *case id* and *activity id* must be selected from the available attributes ($case\_id, activity\_id \in A_{pcs}$) where an attribute can be directly related to an event property without transformations. Given a set of events $E' \subseteq E$, we can compute a multiset of traces $L \in (valueset(activity\_id))^* \rightarrow valueset(case\_id)$ where each trace $\sigma \in L$ corresponds to a case. For example, in Table 1 if we select *sales order* as the *case id*, all events with *sales order = 1* belong to case 1, which can be presented as $\langle A, B, C, D \rangle$. Similarly, case 2 can be presented as $\langle A, C \rangle$. Most control-flow discovery techniques [12–14] use such a simple representation as input. However, the composition of traces can be done using more attributes of events, such as *timestamp, resource* or any other attribute set $A' \subseteq A_{pcs}$.

3 Implementation

This approach has been implemented as a stand-alone java application (available in `http://www.win.tue.nl/~abolt`) named *Process Mining Cube (PMC)* (shown in Fig 6) that has 2 groups of functionalities: *log splitting* and *results generation*. The first consists of creating sublogs (cells) from a large event collection using the operations defined in Sec 2.5, allowing the user to interactively explore the data and isolate the desired behavior of an event collection. The second consists of converting each *materialized cell* into a process mining result, obtaining a collection of results visualized as a 2-D grid, facilitating the comparison between cells. For transforming each *materialized cell* into a process mining result we use existing components and plugins from the ProM framework [17] (`www.processmining.org`) which provides hundreds of plug-ins providing a wide range of analysis techniques.

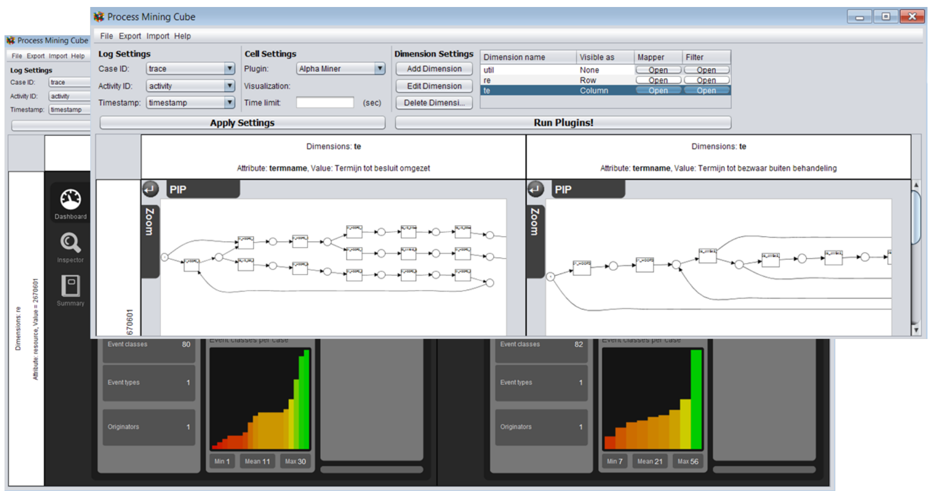


Fig. 6. Process Mining Cube (PMC): Implementation of this approach

The plugins and components used to analyze the cube cells in PMC v1.0 are described in Table 2. This plugin list is extendable. We expect to include more and more plugins for the following versions of PMC.

4 Experiments

In order to compare the performance of our implementation (*PMC*) with the current state of the art (*ProCube*) which was introduced in [4] (also cited in [3]), we designed an experiment using a real life set of events: the WABO1 event log. This log is publicly available in [18] and it is a real-life log that contains

Table 2. Plugins available

Plugin Name	Plugin Description
Alpha Miner	Miner used to build a Petri net from an event log. Fast, but results are not always reliable
Log Visualizer	Visualization that allow us to get a basic understanding of the event log that we are processing
Inductive Miner	Miner that can provide a Petri net or a Process tree as output. Good when dealing with infrequent behavior and large event logs, ensures soundness
Dotted Chart	Visualization that represents the temporal distribution of events
Fast Miner	Miner based on a directly-follows matrix, with a time limit for generating it. The output is a directly-follows graph (Not a ProM plugin)

38944 events related to the process of handling environmental permit requests of a Dutch municipality from October 2010 to January 2014. Each event contains more than 20 data properties. From this property set, only two of them were used as dimensions: *Resource* and *(case) termName*, which produces a 2D cube. Both dimensions were drilled down to its finest-grained level, so every different combination of values from these dimensions creates a different cell.

For both approaches, we compared the *loading time* (e.g. time required to import the events) and *creation time* (e.g. time required to create and materialize all cells and visualize them with the *Log Visualizer* plugin of ProM[17]) using 9 subsets of this log with different number of events. The more events we include in the subset, the larger the value set of a property gets (until the sample is big enough to contain all original values) and more cells are obtained. The experiment results for the 9 subsets are presented in Table 3.

Table 3. Performance benchmark for different-sized subsets of a log

Subset num.		Sub. 1	Sub. 2	Sub. 3	Sub. 4	Sub. 5	Sub. 6	Sub. 7	Sub. 8	Sub. 9
Number of events		1000	5000	10000	15000	20000	25000	30000	35000	38944
Number of cells		48	104	176	187	187	216	216	234	252
ProCube	Load (sec)	2.0	3.0	5.0	8.0	9.0	9.0	9.0	13.0	13.0
	Create (sec)	25.8	106.5	715.3	868.7	1053.2	1220.0	1399.3	1522.5	2279.3
PMC	Load (sec)	0.6	0.9	1.2	1.4	1.6	1.9	2.0	2.1	2.5
	Create (sec)	2.9	6.1	10.1	15.6	21.8	29.5	35.1	41.3	49.6
PMC Load *Speedup*		3.3	3.3	4.1	5.7	5.6	4.7	4.5	6.1	5.2
PMC Create *Speedup*		8.8	17.4	70.8	55.6	48.3	41.3	39.8	36.8	45.9

These results show that *PMC* out-performs the current state of the art in every measured perspective. All *Loading* and *Creation* (Create) times are measured in seconds. Notice that the *Speedup* of *PMC* over *ProCube* is quite considerable, as the average Creation *Speedup* is 40.5 (40 times faster). Also notice that when using the full event log (Sub. 9), *PMC* provides an acceptable response time by creating 252 different process analysis results in less than a minute, something that would take many hours, even days to accomplish if done by hand. This performance improvement makes *PMC* an attractive tool for the academic community and business analysts.

All the above experiments were performed in a laptop PC with an Intel i7-4600U 2.1GHz CPU with 8Gb RAM and Sata III SSD in Windows 7 (x64).

5 Conclusions

As process mining techniques are maturing and more event data becomes available, we no longer want to restrict analysis to a single all-in-one process. We would like to analyse and compare different variants (behaviors) of the process from different perspectives. Organizations are interested in comparative process mining to see how processes can be improved by understanding differences between groups of cases, departments, etc. We propose to use *process cubes* as a way to organize event data in a *multi-dimensional data structure tailored towards process mining*. In this paper, we extended the formalization of process cubes proposed in [3] and provided a working implementation with an adequate performance needed to conduct analysis using large event sets. The new framework gives end users the opportunity to analyze, explore and compare processes interactively on the basis of a multidimensional view on event data. We implemented the ideas proposed in this paper in our *PMC* tool, and we encourage the process mining community to use it. There is a huge interest in tools supporting process cubes and the practical relevance is obvious. However, some of the challenges discussed in [3] still remain unsolved (i.e. comparison of cells and concept drift). We aim to address these challenges using the foundations provided in this paper.

References

1. W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhacement of Business Processes*. Springer-Verlag, Berlin, 2011.
2. J. T. S. Ribeiro and A. J. M. M. Weijters. Event cube: another perspective on business processes. In *Proceedings of the 2011th Confederated international conference on On the move to meaningful internet systems*, Volume Part I, OTM11, pages 274283, Berlin, Heidelberg, 2011. Springer-Verlag.
3. W.M.P. van der Aalst. Process Cubes: Slicing, Dicing, Rolling Up and Drilling Down Event Data for Process Mining. In M.Song, M.Wynn, J.Liu, editors, *Asia Pacific Business Process Management (AP-BPM 2013)*, volume 159 of *Lecture Notes in Business Information Processing*, pages 1-22, Springer-Verlag, 2013.
4. T. Mamaliga. Realizing a Process Cube Allowing for the Comparison of Event Data. Master's thesis, Eindhoven University of Technology, Eindhoven, 2013.
5. W.M.P. van der Aalst, S. Guo, and P. Gorissen. Comparative Process Mining in Education: An Approach Based on Process Cubes. In J.J. Lesage, J.M. Faure, J. Cury, and B. Lennartson, editors, *12th IFAC International Workshop on Discrete Event Systems (WODES 2014)*, IFAC Series, pages PL1.1-PL1.9. IEEE Computer Society, 2014.

6. T. Vogelgesang and H.J. Appelrath. Multidimensional Process Mining: A Flexible Analysis Approach for Health Services Research. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops (EDBT '13)*, pages 17-22, New York, USA, 2013. ACM.
7. S. Chaudhuri, U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD* Rec. 26, pages 6574. 1997.
8. J. Han, M. Kamber. Data mining: concepts and techniques. *The Morgan Kaufmann series in data management systems*. Elsevier, 2006.
9. C. Chen, X. Yan, F. Zhu, J. Han, P.S. Yu. Graph OLAP: a multi-dimensional framework for graph data analysis. Knowledge and Information Systems, vol 21, pages 41-63, Springer, 2009.
10. X. Li, J. Han. Mining approximate top-k subspace anomalies in multi-dimensional time-series data. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)* pages 447-458, VLDB Endowment, 2007.
11. M. Liu, E. Rundensteiner, K. Greenfield, C. Gupta, S. Wang, I. Ari, A. Mehta. E-Cube: Multi-dimensional event sequence processing using concept and pattern hierarchies. In *International Conference on Data Engineering*, pages 1097-1100, 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 International Enterprise Computing Conference (EDOC 2011)*, pages 55-64. IEEE Computer Society, 2011.
13. J.Carmona and J.Cortadella. Process Mining Meets Abstract Interpretation. In J.L. Balcazar, editor, *ECML/PKDD 210*, vol 6321 of *Lecture Notes in Artificial Intelligence*, pages 184-199. Springer-Verlag, Berlin, 2010.
14. 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): pages 215-249, 1998.
15. T. Niemi, M. Niinimäki, P. Thanisch and J. Nummenmaa. Detecting summarizability in OLAP. In *Data & Knowledge Engineering*, vol 89, pages 1-20, Elsevier, 2014.
16. J. Mazón, J. Lechtenbörger and J. Trujillo. A survey on summarizability issues in multidimensional modeling. In *Data & Knowledge Engineering*, vol 68, pages 1452-1469, Elsevier, 2009.
17. B.F. van Dongen, A.K.A. de Medeiros, A.J.M.M. Weijters and W.M.P. van der Aalst. The ProM Framework: A New Era in Process Mining Tool Support. In G. Ciardo, P. Darondeau, editors, *Applications and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444 - 454, Springer, Berlin, 2005.
18. J.C.A.M. Buijs. Environmental permit application process (WABO), CoSeLoG project Municipality 1. Eindhoven University of Technology. Dataset. url: `http://dx.doi.org/10.4121/uuid:c45dcbe9-557b-43ca-b6d0-10561e13dcb5`, 2014.