HAL open science

Comparative Process Mining in Education: An Approach Based on Process Cubes

Wil Aalst, Shengnan Guo, Pierre Gorissen

To cite this version:

Wil Aalst, Shengnan Guo, Pierre Gorissen. Comparative Process Mining in Education: An Approach Based on Process Cubes. 3rd International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA), Aug 2013, Riva del Garda, Italy. pp.110-134. hal-01746404

HAL Id: hal-01746404

https://inria.hal.science/hal-01746404v1

Submitted on 29 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Comparative Process Mining in Education: An Approach Based on Process Cubes

Wil M.P. van der Aalst[1,2], Shengnan Guo[1], and Pierre Gorissen[3]

[1] Eindhoven University of Technology, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
w.m.p.v.d.aalst@tue.nl,s.guo@tue.nl
[2] International Laboratory of Process-Aware Information Systems, National Research University Higher School of Economics (HSE), 33 Kirpichnaya Str., Moscow, Russia.
[3] Fontys Hogescholen, P.O. Box 347, 5600 AH, Eindhoven, The Netherlands
p.gorissen@fontys.nl

Abstract. Process mining techniques enable the analysis of a wide variety of processes using event data. For example, event logs can be used to automatically learn a process model (e.g., a Petri net or BPMN model). Next to the automated discovery of the real underlying process, there are process mining techniques to analyze bottlenecks, to uncover hidden inefficiencies, to check compliance, to explain deviations, to predict performance, and to guide users towards "better" processes. Dozens (if not hundreds) of process mining techniques are available and their value has been proven in many case studies. However, existing techniques focus on the analysis of a single process rather than the comparison of different processes. In this paper, we propose *comparative process mining using process cubes*. An event has attributes referring to the dimensions of the process cube. Through slicing, dicing, rolling-up, and drilling-down we can view event data from different angles and produce process mining results that can be compared. To illustrate the process cube concept, we focus on educational data. In particular, we analyze data of students watching video lectures given by the first author. The dimensions of the process cube allow us to compare the process of students that passed the course versus the process of students that failed. We can also analyze differences between male and female students, between different parts of the course, and between Dutch students and international students. The initial analysis provided in this paper is used to elicit requirements for better tool support facilitating comparative process mining.

Key words: process mining, online analytical processing, learning analytics, 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). As process mining techniques mature, more ambitious types of analysis come into reach. Whereas classical process mining techniques focus on a single process, this paper focuses on *comparing different processes using event data*.

In [3], we proposed 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 that can be used to discover a process model, to check conformance, 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 [20]. 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. male versus female students), subprocesses (e.g. group assignments versus individual assignments), organizational entities (e.g. students versus lecturers), and time (e.g. years or semesters) are semantically different and it is challenging to slice, dice, roll-up, and drill-down process mining results efficiently.

This paper focuses on *comparative process mining using process cubes*. We discuss the main challenges related to comparative process mining. To do this, we use a data set describing behavior of students taking the "Business Information Systems" (2II05) course given at Eindhoven University of Technology. The data set contains two types of events: (1) events generated by students watching recorded video lectures and (2) events generated by students making exams. To understand differences in behavior among different student groups we apply comparative process mining and sketch the possible dimensions of the process cube.

The remainder is organized as follows. In Section 2 we briefly introduce the process mining spectrum. Section 3 defines the process cube notion as a means to view event data from different angles. Then we discuss a concrete case study analyzing the way students watch recorded lectures and correlate this with exam results (Section 4). Section 5 lists the main requirements and open challenges. Related work is briefly discussed in Section 6. Section 7 concludes the paper.

2 Process Mining

Process mining provides a powerful way to analyze operational processes based on event data. Unlike classical purely model-based approaches, process mining is driven by "raw" observed behavior instead of assumptions or aggregate data. Unlike classical data-driven approaches, it is truly process-oriented and relates events to high-level end-to-end process models.

Normally, *event logs* serve as the starting point for process mining. 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.

Fig. 1. Dotted chart showing all events related to the course Business Information Systems (2II05). The dotted chart was created using one of the over 600 *ProM* plug-ins (cf. www.processmining.org).

Often event logs store additional information about events, e.g., the *resource* (i.e., person or device) executing or initiating the activity, the *timestamp* of the event, or *data elements* recorded with the event.

Process mining has been applied in a wide variety of organizations (e.g., banks, hospitals, municipalities, governmental agencies, webshops, and high-tech system manufacturers). Moreover, there are dozens of process mining techniques answering a wide variety of questions [1]. Due to space restrictions we can only illustrate a fraction of the available process mining techniques. To do so, we use a concrete data set involving two types of events recorded for students that took the Business Information Systems (2II05) course at Eindhoven University of Technology from 2009 until 2014. A *view* event refers to a student watching a particular lecture. An *exam attempt* event refers to a student taking an exam. Since the course is quite challenging, it is not uncommon that students need to resit the 2II05 exam multiple times. There are at least two exams per year. The initial log contains 6744 events generated by 287 students.

Figure 1 shows a so-called *dotted chart* taking the viewpoint that each student corresponds to a case (i.e., process instance). The dotted chart has 287 horizontal lines each corresponding to a student. The dots along such a line define the corresponding trace. The color of the dot refers to the corresponding activity, e.g., viewing a particular lecture. The red dots in Figure 1 refer to exam attempts. It can be seen that some students need to take multiple exams and that students

tend to watch irregularly. Note that the video lectures are an additional service to the students (i.e., next to regular lectures).

Let us zoom in on the group of 47 students taking the course in the period January 2011 - August 2011. These students took the exam on 21-6-2011 and/or the retake exam on 16-8-2011. There were 24 lectures (two lectures per week) in the period from January until May 2011. Students could watch the lectures via an internet connection soon after recording. For example, activity "2II05 College 11b" refers to the second lecture in the 11th week. Figure 2 shows a process model discovered for this event log. Indeed students tend to watch the videos in chronological order. However, Figure 2 only considers the most frequent paths. The actual process is more "Spaghetti-like".

We have made a process model having 24 viewing activities in sequence followed by an exam. Hence, the model only allows for the trace ⟨ "2II05 College 1a", "2II05 College 1b", "2II05 College 2a", "2II05 College 2b", ..., "Exam"⟩. Using *ProM* we can check the conformance of such a model in various ways. For example, we can compute alignments that map the traces in the event log to valid paths in the model [5, 10]. Figure 3 shows four such alignments. Overall, the fitness is low (0.33) showing that despite the tendency to watch lectures in the expected order, few students do so consistently. Also the times at which students watch the videos show a lot of variation. For example, it can be noted that, other than just before the exam, students rarely watch video lectures in the second half of the course.

3 Process Cubes

Figures 1, 2 and 3 show some example results computed for a given event log, i.e., a collection of events. However, often the goal is to compare different variants of the same process or to zoom in on particular parts or aspects of the process. OLAP (Online Analytical Processing) tools aim to support this by organizing data in a cube having multiple dimensions [20]. However, OLAP is only used for numerical comparisons, e.g., the average transaction amount in different shops on different days in the week. In a *process cube* [3] we organize events using different dimensions and compute a (process) model per sublog associated to a cell. This way we can slice, dice, roll-up, and drill-down process mining results easily.

Throughout the paper we assume the following universes [3].

Definition 1 (Universes). *$\mathcal{U}_V$ is the universe of possible attribute values (e.g., strings, numbers, etc.). $\perp \in \mathcal{U}_V$ denotes a missing ("null") value. $\mathcal{U}_S = \mathcal{P}(\mathcal{U}_V)$ 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., $V = \{male, female\}$), and $H \in \mathcal{U}_H$ is a collection of sets. For example, $H = \{\{x \in \mathbb{N} \mid x < 50\}, \{x \in \mathbb{N} \mid 40 \leq x < 80\}, \{x \in \mathbb{N} \mid x \geq 70\}\}$.

An *event base* is a "raw" collection of *events* having *properties*.

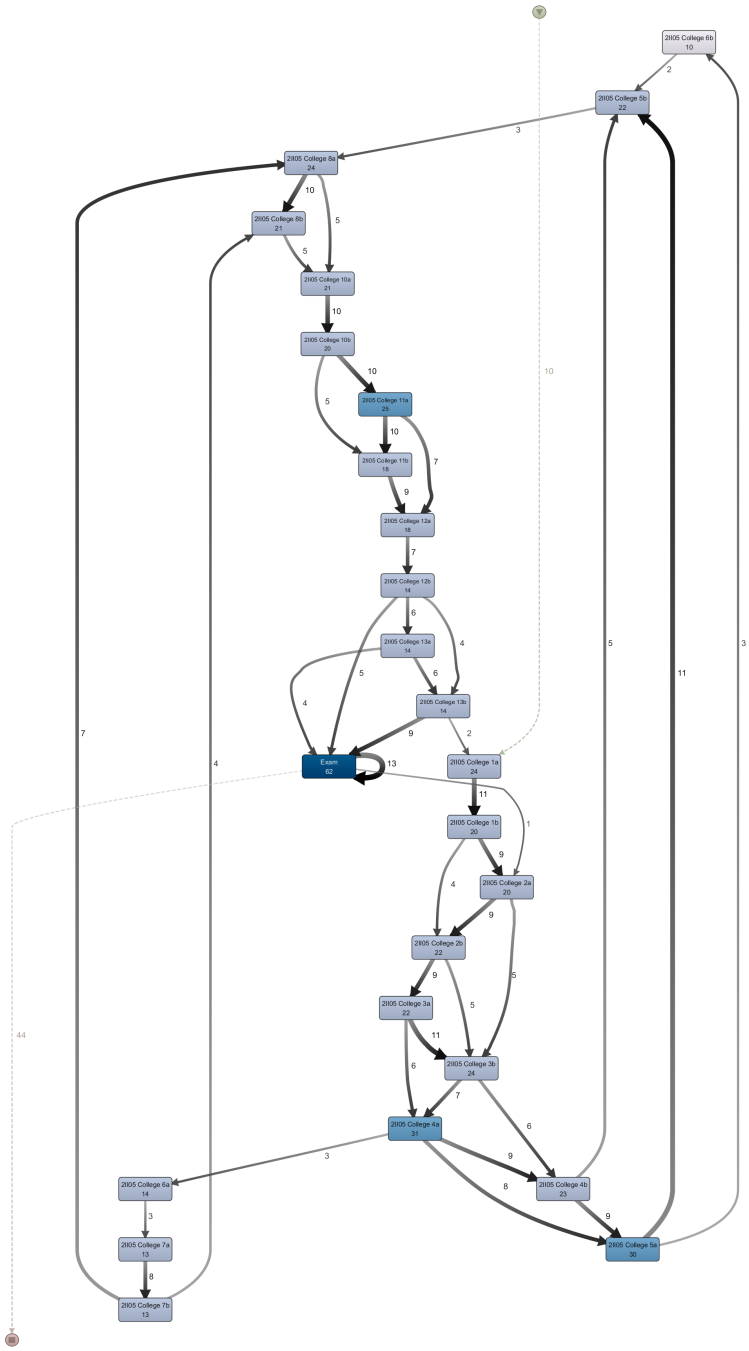

Fig. 2. Process model created using *Disco* while abstracting from infrequent paths. There are 24 activities corresponding to the video lectures and one activity corresponding to an exam attempt. The coloring indicates the frequency of the activities.

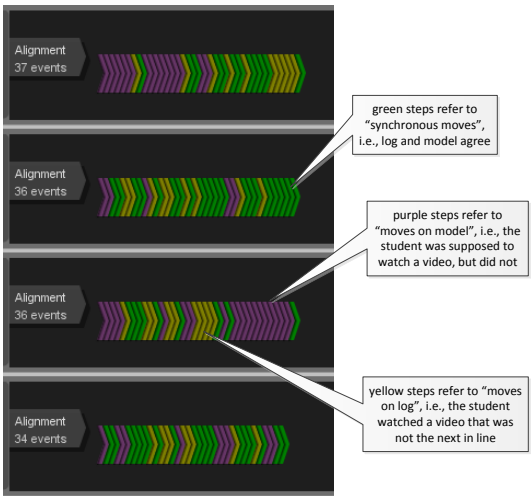

Fig. 3. Alignments for four students showing that few students actually watch the lectures sequentially. The "synchronous moves" show where model and log agree. The "moves on model" show where the student skipped a video lecture. The "moves on log" show where the student watched a video lecture that was not next in line.

Definition 2 (Event Base, [3]). *An* event base $EB = (E, P, \pi)$ *defines a set of* events E*, a set of* event properties P*, and a function* $\pi \in P \rightarrow (E \rightarrow \mathcal{U}_V)$*. For any property* $p \in P$*,* $\pi(p)$ *(denoted* π_p*) is a function mapping an event onto a value for property* p*. If* $\pi_p(e) = v$*, then event* $e \in E$ *has a property* $p \in P$ *and the value of this property is* $v \in \mathcal{U}_V$*. We write* $\pi_p(e) = \perp$ *for missing values.*

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, [3]). *A* process cube structure *is a triplet* $PCS = (D, type, hier)$ *where:*

- D *is a set of dimensions,*
- $type \in D \rightarrow \mathcal{U}_S$ *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*
- $hier \in D \rightarrow \mathcal{U}_H$ *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. To relate an event base and a process cube structure, both need to be *compatible*, i.e., dimensions should correspond to properties ($D \subseteq P$) and concrete event property values need to be of the right type ($\pi_d(e) \in type(d)$ for any $d \in D$ and $e \in E$). Moreover, for process mining we often assume that $\{case, activity, time, resource\} \subseteq D \subseteq P$, i.e., each event refers to a case and an activity, occurred at a particular time, and was executed by a particular resource. These properties do not need to be

decided upfront and can be changed during analysis. For example, the notion of *case* may be changed to create another viewpoint (see Chapter 4 in [1]).

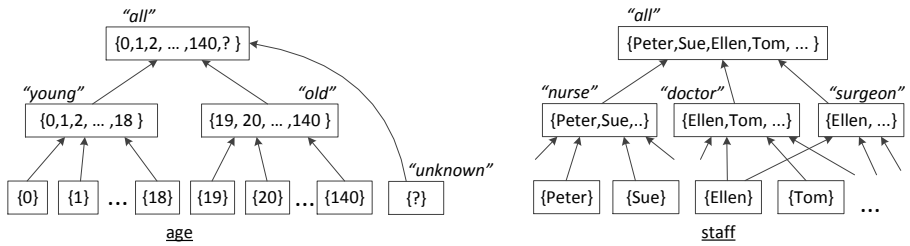

Fig. 4. Two example hierarchies, both having three levels. The left-hand-side hierarchy can be used to group events according to the patient's age. The right-hand-side hierarchy can be used to group events according to the role of the resource performing the activity.

To clarify the notion of a *process cube structure*, let us consider an example $PCS = (D, type, hier)$.

- $D = \{patient_id, type, age, activity, staff, time, \ldots\}$ defines the set of dimensions.
- *type* is a function mapping each dimension onto a set of possible values:
  - $type(patient_id) = \{99000, 99001, \ldots 99999\}$ is the set of patient identifiers (this value *can* be used as a case identifier),
  - $type(type) = \{gold, silver\}$ is the set of patient types (patients of type *gold* have a better insurance allowing for extra privileges),
  - $type(age) = \{0, 1, 2, \ldots, 140, ?\}$ is the set of possible ages (value "?" denotes that the age of the patient is unknown),
  - $type(activity) = \{blood_test, doctor_visit, X_ray, handle_payment, \ldots\}$ is the set of activities,
  - $type(staff) = \{Peter, Sue, Ellen, Tom, \ldots\}$ is the set of resources (doctors, nurses, and
  - $type(time)$ is the set of possible timestamps.
- *hier* is a function defining a hierarchy for each dimension:
  - $hier(opatient_id) = \{\{99000, 99001, \ldots 99999\}, \{99000\}, \{99001\}, \{99002\}, \ldots, \{99999\}\}$. The element $\{99000, 99001, \ldots, 99999\}$ can be seen as the root of the hierarchy, i.e., all possible values. The other singleton elements can be seen as the leaves of the hierarchy, e.g., the set $\{99023\}$ represents one individual patient.
  - $hier(type) = \{\{gold, silver\}, \{gold\}, \{silver\}\}$. Set $\{gold, silver\}$ can be seen as the root of the hierarchy, i.e., all patient types. The singleton $\{gold\}$ refers to patients having special privileges and the singleton $\{silver\}$ refers to "normal" patients.

- $hier(age) = \{\{0, 1, 2, \ldots, 140, ?\}, \{0, 1, 2, \ldots, 18\}, \{19, 20, \ldots, 140\}, \{0\}, \{1\}, \{2\}, \ldots, \{140\}, \{?\}\}$ defines an age hierarchy with three levels (see Figure 4), e.g., the element $\{0, 1, 2, \ldots, 18\}$ represents the subset "young".
- $hier(activity) = \{\{blood_test, doctor_visit, X_ray, handle_payment, \ldots\}, \{blood_test\}, \{doctor_visit\}, \ldots\}$ groups all activities in a simple hierarchy,[1]
- $hier(staff) = \{\{Peter, Sue, Ellen, Tom, \ldots\}, \{Peter, Sue, \ldots\}, \{Ellen, Tom, \ldots\}, \{Ellen, \ldots\}, \{Peter\}, \{Sue\}, \{Ellen\}, \{Tom\}, \ldots\}$ defines a role hierarchy as shown in Figure 4. Staff member *Ellen* is both a doctor and a surgeon and appears in four subsets of $hier(staff)$.
- $hier(time)$ can be used to group timestamps into years, months, weekdays, days, etc.

Figure 4 shows two hierarchies. The arcs are based on inclusion and can be interpreted as "is a". In this paper we use a rather simplistic, but also general, notion of hierarchy. Only the possible elements of a dimension are mentioned and no further constraints are given. We do not specify which elements can be used at the same time (see also [3]). Normally, one will make the different levels explicit and only select elements of a given level. We also do not specify navigation rules and do not explicitly name the sets in a hierarchy. The names used Figure 4 ("all", "young", and "nurse") are not formalized, but should of course be supported by software tools.

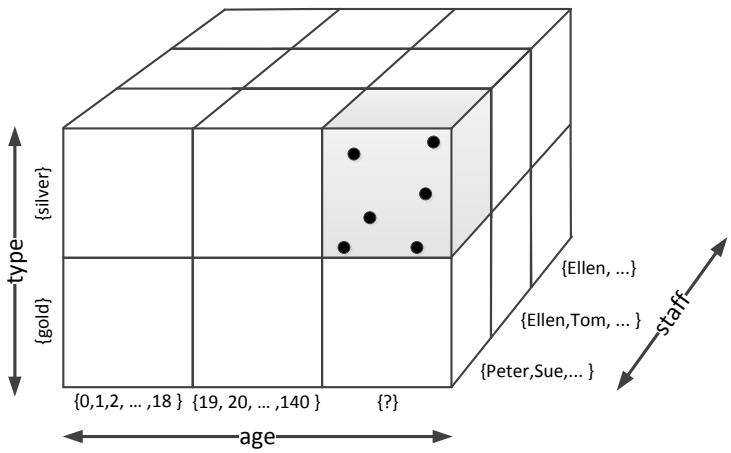

Fig. 5. Process cube view having three selected dimensions: $D_{sel} = \{type, age, staff\}$.

A *process cube view* defines which dimensions are visible and which events are selected.

[1] The $hier(activity)$ hierarchy could be further refined based on some taxonomy. Here we just consider individual activities (singleton sets) and the set of all activities.

Definition 4 (Process Cube View, [3]). *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,*
- *$sel \in D \rightarrow \mathcal{U}_H$ is a function selecting the part of the hierarchy considered per dimension. Function sel is such that for any $d \in D$: $sel(d) \subseteq hier(d)$.*

Figure 5 shows an example of a process cube view. $D_{sel} = \{type, age, staff\}$ are the selected dimensions. $sel(type) = \{\{gold\}, \{silver\}\}$, $sel(age) = \{\{0, 1, 2, \ldots, 18\}, \{19, 20, \ldots, 140\}, \{?\}\}$, and $sel(staff) = \{\{Peter, Sue, \ldots\}, \{Ellen, Tom, \ldots\}, \{Ellen, \ldots\}\}$. Note that function *sel* returns a set of sets for each dimension. For the non-selected dimensions $D \setminus D_{sel}$ we cannot see the value of *sel* in Figure 5, but still these dimensions may have been used for filtering. For example, it may be that $sel(activity) = \{\{blood_test\}\}$, i.e., only events related to blood tests are considered. Moreover, it could be that $sel(type)$ is used to select events that happened in 2013 and 2014.

By removing elements from D_{sel}, the number of dimensions is reduced. This is orthogonal to function *sel* which decides on the granularity and filtering of each dimension. Given an event base and a process cube view, we can compute an event log for every cell in the process cube.

Definition 5 (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 $cells = \{c \in D_{sel} \rightarrow \mathcal{U}_S \mid \forall_{d \in D_{sel}}\ c(d) \in sel(d)\}$ 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.*

The term "materialized" may be misleading, it is just used to express that, in order to apply standard process mining techniques, we need to create an event log (e.g., in XES format [28]) for every cell in the process cube view. Figure 5 highlights one of the 18 cells: This cell contains all events corresponding to a "normal" patient having a unknown age, and performed by a nurse. The events in this cell can be transformed into an event log and analyzed using process mining techniques. Moreover, results for different cells can be compared.

Using the above definition we can formalize notions such as slice, dice, roll-up, and drill-down for process cubes.

Definition 6 (Slice, [3]). *Let $PCS = (D, type, hier)$ be a process cube structure and $PCV = (D_{sel}, sel)$ a view of PCS. For any $d \in D_{sel}$ and $V \in sel(d)$: $slice_{d,V}(PCV) = (D'_{sel}, sel')$ with $D'_{sel} = D_{sel} \setminus \{d\}$, $sel'(d) = \{V\}$, and $sel'(d') = sel(d')$ for $d' \in D \setminus \{d\}$.*

Through slicing, a dimension d is removed. At the same time one value set V is chosen for the removed dimension.

For example, Figure 6 shows $slice_{staff,\{Peter,Sue,\ldots\}}(PCV)$, starting from the view in Figure 5. This new process cube view is the result of slicing using dimension *staff* and value set $\{Peter, Sue, \ldots\}$, i.e., the *staff* dimension is no longer

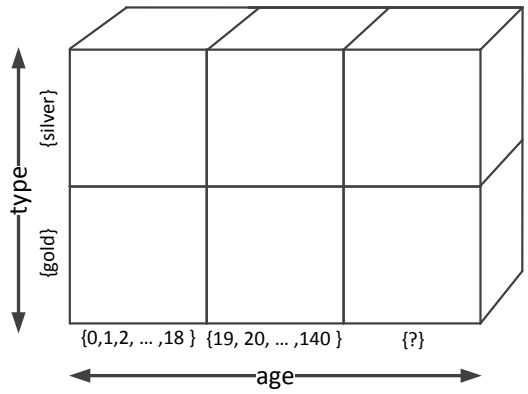

Fig. 6. The process cube view after slicing based on dimension *staff*.

selected and only events performed by nurses are considered in the resulting view.

Definition 7 (Dice, [3]). *Let $PCS = (D, type, hier)$ be a process cube structure and $PCV = (D_{sel}, sel)$ a view of PCS. Let $res \in D_{sel} \nrightarrow \mathcal{U}_H$ be a restriction such for any $d \in dom(res)$: $res(d) \subseteq sel(d)$. $dice_{res}(PCV) = (D_{sel}, sel')$ with $sel'(d) = res(d)$ for $d \in dom(res)$ and $sel'(d) = sel(d)$ for $d \in D \setminus dom(res)$.*

Dicing does not remove a dimension, but restricts the values sets for one or more dimensions.

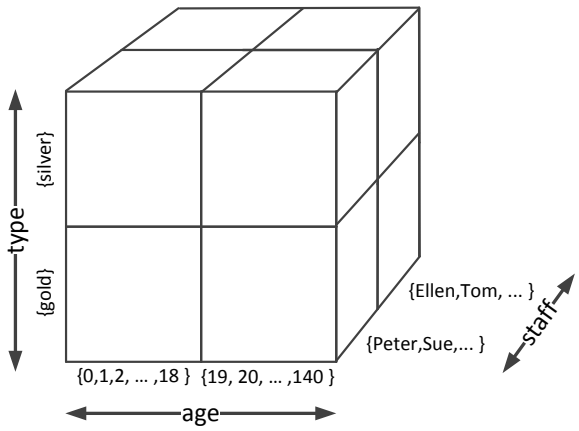

Fig. 7. The result after dicing based on dimensions *staff* and *age*.

Consider Figure 5 again. Suppose that we would like to remove events referring to patients whose age is unknown and that we only consider events per-

formed by nurses and doctors. Hence, $dom(res) = \{staff, age\}$ because these are the two dimensions we would like to dice. Moreover, $sel(age) = \{\{0, 1, 2, \ldots, 18\}, \{19, 20, \ldots, 140\}\}$ (note that $\{?\}$ was removed) and $sel(staff) = \{\{Peter, Sue, \ldots\}, \{Ellen, Tom, \ldots\}\}$ (note that the surgeon role was removed). The result is shown in Figure 7.

Definition 8 (Change Granularity, [3]). *Let $PCS = (D, type, hier)$ be a process cube structure and $PCV = (D_{sel}, sel)$ a view of PCS. Let $d \in D_{sel}$ and $H \in \mathcal{U}_H$ such that: $H \subseteq hier(d)$ and $\bigcup H = \bigcup sel(d)$. $chgr_{d,H}(PCV) = (D_{sel}, sel')$ with $sel'(d) = H$, and $sel'(d') = sel(d')$ for $d' \in D \setminus \{d\}$.*

Drilling down is an example of an operator for changing the granularity of the cube, e.g., refining a year into months. However, it is also possible to make the view more coarse (roll-up).

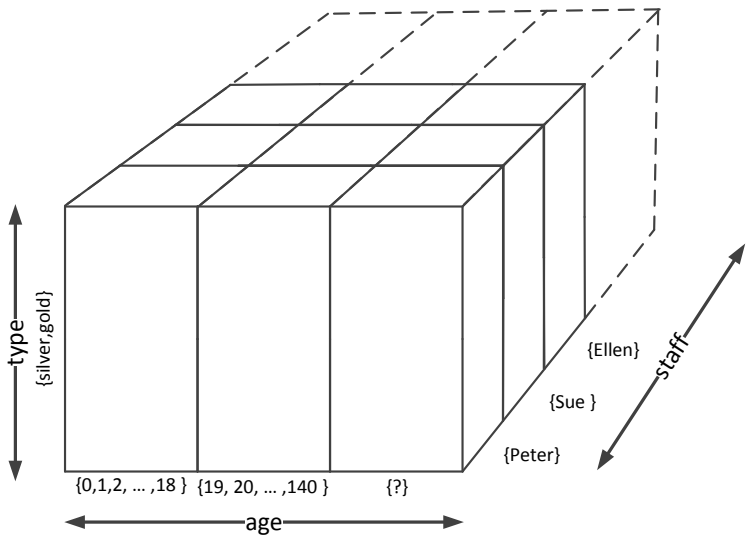

Fig. 8. The process cube view after changing the granularity of dimensions *type* and *staff*. The *type* dimension was rolled up (made coarser) and the *staff* dimension was refined (drill-down to the level of individual staff members).

Figure 8 shows a process cube view that was obtained by changing granularity of the original view depicted in Figure 5. Function $chgr_{d,H}$ was applied twice: the *type* dimension was coarsened and *staff* dimension was refined. Events related to silver and gold patients have been merged, but still the *type* dimension is visible. Moreover, events are now related to individual staff members rather than roles. Figure 8 now uses the leaves of the *staff* hierarchy.

Through slicing (cf. Figure 6), dicing (cf. Figure 7), and changing the granularity (cf. Figure 8), we can change the process cube view in Figure 5. At any

point in time we can generate an event log per cell and compare the process mining results. To be able to apply process mining per cell, the classical requirements need to be satisfied, i.e., events need to be (partially) ordered (e.g., based on some timestamp), one needs to select a case identifier to correlate events and an event classifier to determine the activities. See [1, 3, 28] for more information on process mining, process cubes, and event logs.

Based on the definitions in this section we have developed an initial prototype (called *ProCube*) using the process mining framework *ProM* and the *Palo OLAP* toolset [31]. *ProCube* runs as a plugin in *ProM*. The plug-in creates sublogs per cell on-the-fly and visualizes process models discovered using the fuzzy miner [26] and the heuristic miner [45], social networks derived using *ProM*'s social network miner [7], and dotted charts [42] computed per cell. The prototype has many limitations (too slow for high-dimensional process cubes, poor visualization of the results, and limited support for hierarchies), but nicely illustrates the concepts. Currently, we are working on more mature process cube support through software.

4 Video Lectures: A Case Study

The primary goal of the process cube notion is to facilitate comparison. To illustrate this, we return to the data set described in Section 2. At Eindhoven University of Technology various lectures are recorded and made available online. The *Lecture Capturing System* (LCS) used is *Mediasite* developed by SonicFoundry. This system is able to provide an audit trail, but does not provide any form of process mining. We also used a database with exam results to relate student performance to viewing behavior. Student names were replaced by anonymous identifiers before analysis. Our analysis builds on the PhD research of Pierre Gorissen who analyzed the viewing behavior of students using Mediasite by means of more traditional methods rather than process mining [25].

4.1 Data Available on Video Lectures and Exams

To understand the data available consider the class diagram shown in Figure 9. A *course* has a unique code and a name. The same course may be given multiple times, e.g., once or twice per year. Such a *course instance* has a start date and an end date, in-between these dates *video lectures* are recorded. Per course there are *exams*. Each exam refers to the *last* course instance given. Per course instance there are one or more corresponding exams (typically a regular exam and an additional exam for the students that failed). A *student* may use multiple *exam attempts*. Such an attempt refers to an exam on a particular day and a student. Students may view lectures, also of earlier course instances. An *atomic view* refers to one student and one lecture. Per atomic view the interval watched is recorded, e.g., a student watches the first 14 minutes of "2II05 College 11b". Also the time and date at which the student watches the fragment is recorded.

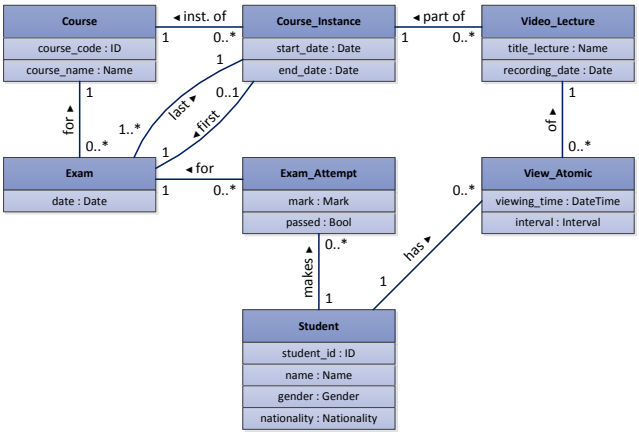

Fig. 9. Overview of the raw data available.

4.2 Identifying Events

Students can view a lecture in smaller chunks, e.g., a student can fast forward 20 minutes, then watch 5 minutes, go back to be beginning and watch 10 minutes, etc. Therefore, the same student may generate hundreds of atomic views of the same lecture on the same day. Since we would like to have a model at the level of lectures, we add a new class named "View (derived)" in Figure 10. Entities of this class correspond to compositions of atomic views. Attribute "nof_views" in Figure 10 refers to the number of atomic events and "total_duration" is the sum of all the durations of the corresponding atomic events. We also record the start time of the first and last atomic view of the lecture by that student on that day.

Based on the available data we propose two types of events: *exam attempts* (entities of class "Exam_ Attempt") and *views* (entities of class "View (derived)"). Figure 10 shows all the properties of these events. These properties follow directly from the class model. For example, a view refers to a student and her properties (id, name, gender, and nationality), a view refers to a video lecture and its properties (name and recording date). Because a video lecture belongs to a course instance and therefore also to a course, additional properties can be derived.

4.3 Defining the Process Cube Structure and Selecting Views

The two events types in Figure 10 list many properties. When merging both event types in an event base $EB = (E, P, \pi)$ we take the union of these. Each of the properties may serve as a dimension in the process cube structure $PCS = (D, type, hier)$. If a course is composed of different parts, function *hier* can be used to reflect this. Also there is a natural hierarchy for the time domain (years, months, weeks, days, etc.). It is also possible to group exams in a hierarchical

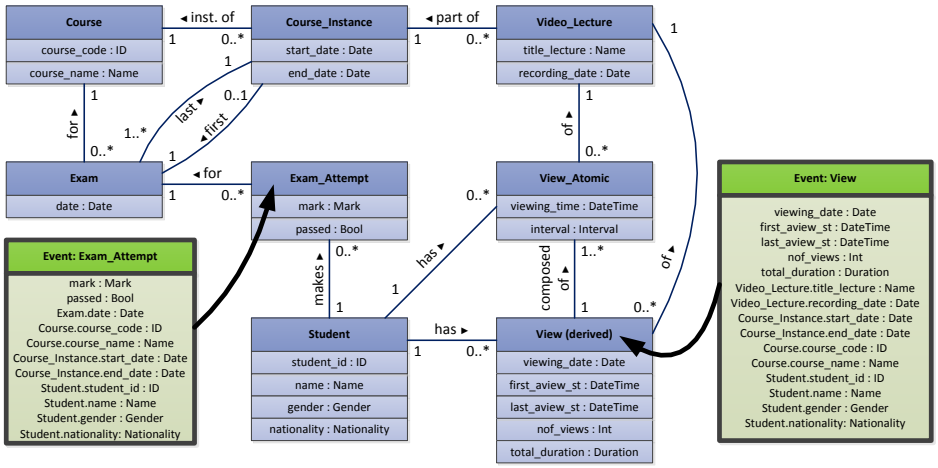

Fig. 10. The class diagram with the two types of events considered: *views* and *exam attempts*.

manner: a course instance or course can also be viewed as a collection of exams. This way we obtain a hierarchy consisting of three levels: individual exams, exams belonging to a course instance, and exams for a course. The video lectures can also be grouped in a hierarchical manner.

The process cube view $PCV = (D_{sel}, sel)$ defines the cells that we would like to consider. We can slice the cube to focus on a particular course. For example we can select only events related to course "2II05" and remove the dimensions "Course.course_code" and "Course.course_name" from D_{sel}. The process cube operators defined in Section 3 can be used to create the desired view $PCV = (D_{sel}, sel)$. At any time the view PCV can be materialized resulting in an event log per cell (sublogs). Figure 11 illustrates the overall process. *Once we have an event log per cell, we can apply any process mining technique to all cells and compare the results.* This facilitates comparative process mining.

Recall that for process mining we often assume that $\{case, activity, time, resource\} \subseteq D$. $\pi_{case}(e)$ is the case associated to event e. There are two obvious choices: a case is an exam attempt or a case is a student. If a student used three exam attempts and we assume the first notion, then the same student will generate three cases. Also other choices are possible, e.g., cases may also correspond to video lectures or course instances. To simplify the interpretation of the results in this paper, we assume that $\pi_{case}(e)$ equals the "Student.Student_id" event attribute in Figure 10. For exam events we choose $\pi_{activity}(e)$ to be the string "Exam". For view events we choose $\pi_{activity}(e)$ to be the title of the lecture, e.g., "2II05 College 11b". For exam events we choose $\pi_{time}(e)$ to be the date and time of the exam. For view events we choose $\pi_{time}(e)$ to be the date and time of viewing. We actually have start and complete events for all activities and

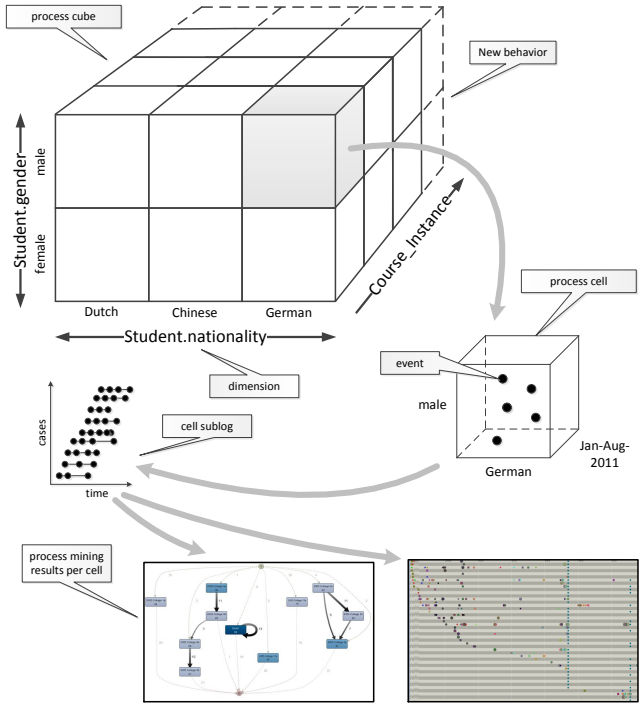

Fig. 11. Given a process cube view, the cells can be materialized. Subsequently, process mining techniques can be applied to the cell sublogs.

can thus measure the duration of activities. Resources seem less relevant here because the case is a student and hence $\pi_{resource}(e)$ refers to the case itself.

4.4 Analyzing Process Cube Views: Some Examples

Assume we have "sliced and diced" the process cube in such a way that we only consider the course instance of 2II05 running from January 2011 until August 2011. Moreover, D_{sel} contains only the dimensions *gender* (male of female) and *nationality* (Dutch or international). This results in four cells. Table 1 shows some results for this process cube view. Dutch students tend to watch the video lectures more frequently and are more likely to pass. Students that pass tend to watch the video lectures more frequently than the ones that do not pass. Note that Table 1 is based on just the 47 students that followed this particular course instance. Hence, based on these findings we cannot (and should not) generalize.

Table 1 also shows the average trace fitness for each of the four cells. Conformance checking based on alignments and a sequential idealized process model are used to compute these numbers (see Section 2). The average trace fitness is 1 if all the students watch all video lectures in the default order and conclude

with an exam. The Dutch students that passed have the highest average trace fitness (0.39). International students and students that failed have a lower average trace fitness. The average trace fitness for all students that passed is 0.37. This is significantly higher than the average trace fitness for all students that did not pass (which is 0.28).

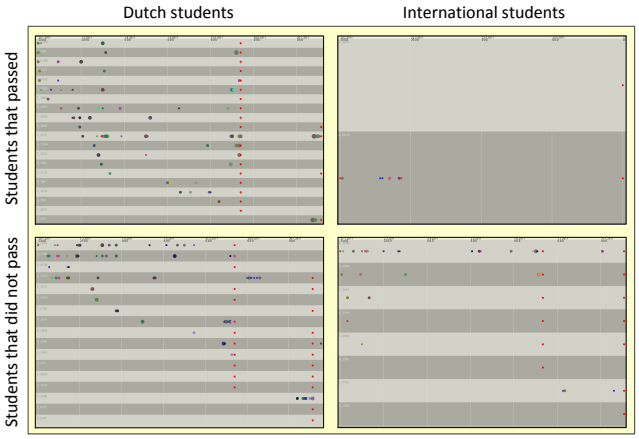

Fig. 12. Four dotted charts based on the simple process cube view having two dimensions.

Any process mining algorithm can be applied to the materialized cells of a process cube view. Figure 12 shows dotted charts for each of the four cells also used in Table 1. It shows that the Dutch students that passed often took the first exam and passed immediately. They also watched the lectures right from the start. The students that did not pass often skipped the first exam or even made

Table 1. Some numerical results based on a simple process cube view having only two selected dimensions.

	Dutch student	international student
student passed	number of students: 20 events per student: 28.5 views per student: 13.1 exams per student: 1.15 trace fitness: 0.39	number of students: 2 events per student: 12 views per student: 4 exams per student: 2 trace fitness: 0.25
student did not pass	number of students: 17 events per student: 20.2 views per student: 8.7 exams per student: 1.4 trace fitness: 0.29	number of students: 8 events per student: 17.5 views per student: 7.1 exams per student: 1.6 trace fitness: 0.26

two unsuccessful attempts. It can also be seen that some students systematically watched the videos to prepare for the exam.

For each cell we can also discover process models using process mining techniques. Here, we compare the students that passed (Figure 13) with the students that did not pass (Figure 14), i.e., we only use one dimension for comparison. Figure 13 is based on 22 cases and Figure 14 is based on 25 cases. Hence, it is not easy to generalize the results. However, there are obvious differences between both process models that confirm our earlier findings. Students that pass, tend to watch the lectures more regularly. For example, students that pass the course tend to start by watching the first lecture (see connection from the start node to "2II05 College 1a"), whereas students that fail tend to start by making the exam (see connection from the start node to "Exam") rather than watching any video lectures.

Although the small of number students in the selected course instance does not allow for reliable generalizations, the case study nicely illustrates the applicability of process cubes for comparative process mining. Due to space restrictions, we could only present a fraction of the results and touched only a few of the available dimensions. For example, we also compared different courses instances, investigated differences in study behavior between male and female students, etc.

5 Requirements and Challenges

The case study just presented nicely illustrates the usefulness of process cubes for comparative process mining. However, the case study also reveals severe limitations of the current approach and implementation.

5.1 Performance

The process cube notion is most useful if the users can *interactively* "play with the cube" (slice, dice, drill-down, roll-up, etc.). However, this necessitates a good *performance*. There are two potential performance problems: (1) it takes too long to materialize the cells selected for analysis (i.e., create the event logs) and (2) it takes too long to compute the process mining results for all selected cells. Most process mining algorithms are linear in the number of cases and exponential in the number of activities or average trace length. Hence, it may be worthwhile to precompute results. However, if there are many dimensions each having many possible values, then this is infeasible. In the latter case one also needs to deal with the *sparsity* problem [36]. Suppose that there are just 10 dimensions each having just 10 possible values. Then, there are 10^{10} cells at the lowest level of granularity. This means that even if we have one million of events, at least 99.99% of cells will be empty. This creates an enormous overhead if sparsity is not handled well by the implementation [31].

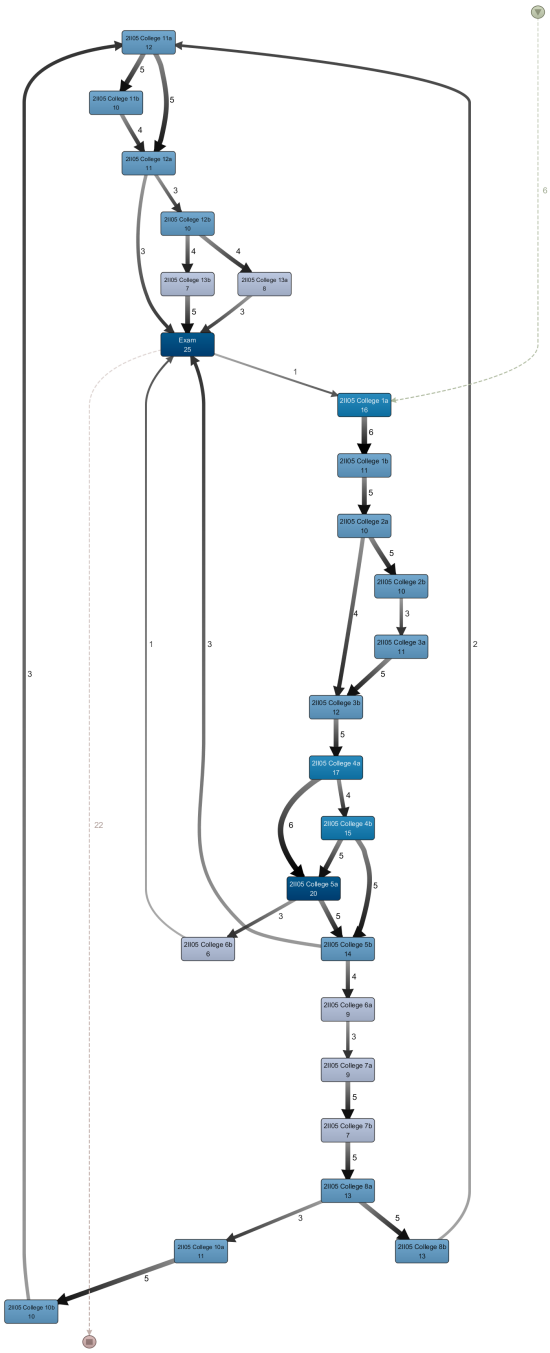

Fig. 13. Process model for the students that passed.

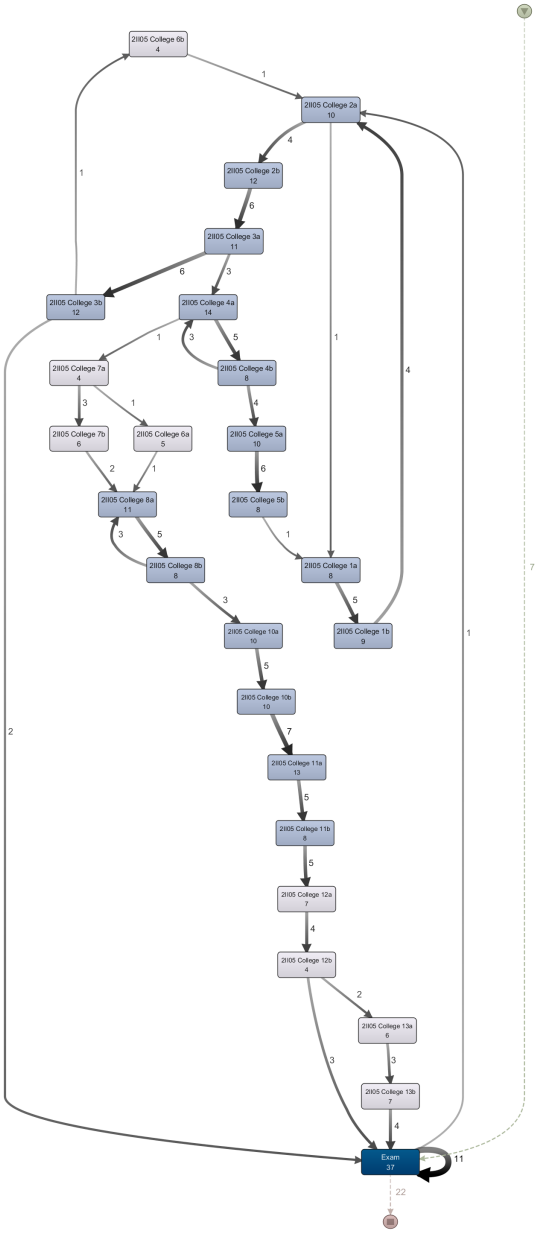

Fig. 14. Process model for the students that did *not* pass.

5.2 Interpreting the Results: Comparing Graphs

Another difficulty is the *problematic interpretation* of the results. The goal of showing multiple process mining results is to facilitate comparison. However, this is far from trivial as is illustrated by the four dotted charts in Figure 12. How to highlight differences and commonalities among multiple dotted charts?

Process mining results are often presented as *graphs*. In earlier sections we presented a few discovered process models showing only the control-flow. Most discovery algorithms return process models that are graphs (Petri nets, BPMN models, transition systems, Markov chains, etc.). Moreover, many other types of mining algorithms create graphs, e.g., organizational graphs, social networks, or richer process maps also showing data flow and work distribution [1]. The layout of such graphs is typically not tailored for comparison. See for example Figure 13 where the "Exam" activity appears in the upper half of the diagram and Figure 14 where the same activity appears at the bottom of the process map. To visualize such networks the nodes in the different graphs need to be aligned. Assuming that there is a mapping between the nodes of the graphs, e.g., based on activity labels, there are different ways of visualizing such graph alignments. One can use a "side-by-side" approach where the individual graphs are shown next to each other. To highlight the aligned nodes one can try to give related nodes the same relative position. One can also create a "all-in-one" graph that is the union of all individual graphs and use coloring or annotations to facilitate the reconstruction of the individual graphs. [15] propose a mixture of the above two approaches (2.5D layouts). Here, the correspondence of aligned nodes is implied by drawing all 2D layouts simultaneously. The third dimension is used to place corresponding nodes on top of each other. The all-in-one approach seems to be most promising for a binary comparison of two models. Here one can take one of the two graphs as a reference and then highlight the differences in a so-called comparison graph. For example, if the arcs in the individual graphs are annotated with durations, then the comparison graph can show the differences highlighting parts that are faster or slower than the reference process.

5.3 Refinements

Next to challenges in performance and visualization, also conceptual refinements of the process cube notion are needed. We use the data set presented in Section 4 to illustrate these.

As Figure 10 indicates, we have two very distinct types of events (exam attempts and views) having different properties. By taking the union over these properties, we get many missing values (i.e., $\pi_p(e) = \perp$ for property p and event e). For example, an exam attempt event does not have a "nof_views" property and a view event does not have a "mark". Moreover, one should make sure that the properties having the same name across different event types actually have the same semantics. For process mining, it makes no sense to only focus on events of one particular type. End-to-end processes may refer to a wide variety of activities. To address this problem one could extend the process cube notion

into an *array of process cubes* (one for each type of event) with explicit relations between them. Alternatively, one can also define dimensions to be irrelevant for certain event types. For example, when slicing the process cube using the "mark" dimension (e.g. $slice_{mark,\{8\}}(PCV)$) one would like to retain all view events and only remove exam attempts having a different mark (e.g. not equal to 8). In the current formalization this can be mimicked by including events having a $\perp$ value when doing a slice or dice action.

When applying process mining, events need to refer to a case and an activity, and have a timestamp. Hence, $\{case, activity, time\} \subseteq D$. However, these are *not* ordinary dimensions. For example, we may alter the notion of a case during analysis. For instance, using the event data described in Figure 10, we can first consider students to be the cases and later change the case notion into exam attempts. We can investigate the study progress of a group of students across different courses (i.e., a case should refer to a student). We can also investigate the results for a particular exam (i.e., a case refers to an exam attempt). Similarly, we can change the activity notion. For example, in our analysis we did not distinguish between different exams when naming activities ($\pi_{activity}(e) =$ "Exam"). We could also have opted for different activity names depending on the exam (date and/or course). The process cube should support various case and activity notions and not consider these to be fixed upfront.

The same event may appear in *multiple* cells, e.g., organizational units may partially overlap and subprocesses may have common interface activities [3]. The notion of hierarchy should be as general as described in Section 3 to allow for this.

A case is defined as a collection of events. Hence, a case cannot have zero events, because it will simply not appear in the process cube. This may lead to incorrect or misleading interpretations. On the one hand, some process mining algorithms cannot handle empty traces. On the other hand, also empty traces contain information. See for example the importance of empty traces in decomposed process mining [2]. Now all dimensions in PCV are event properties. However, *some properties clearly reside at the case level.* For example, a view event in Figure 10 has as property the gender of the student that viewed the lecture. However, this is more a property of the case (i.e., student) rather than the event (i.e., view). As a result, information is replicated across events. Hence, it may be useful to distinguish between case dimensions and event dimensions in a process cube. Moreover, it is often useful to exploit *derived* case information. For example, we may want to use the number of failed exam attempts or flow time. Generally speaking processes are executed within a particular context [6], and events and cases may need to be enriched with derived context information. For an event e, we may want to know the utilization level of the resources involved in the execution of e, and insert this utilization level as another dimension. For a case, we may want to know the relative duration compared to other cases and insert it as a dimension at the case level.

6 Related Work

See [1] for an introduction to process mining and the Process Mining Manifesto [27] for the main challenges in process mining. For example, dozens of process discovery [1, 8, 9, 13, 23, 14, 18, 19, 21, 24, 33, 41, 29, 30, 45, 46] and conformance checking [5, 10, 11, 12, 16, 22, 24, 34, 35, 39, 44] approaches have been proposed in literature. This paper is not about new process mining techniques, but builds on the notion of process cubes introduced in [3]. Process cubes are related to the well-known OLAP (Online Analytical Processing) cubes [20] and large process model repositories [38].

7 Conclusion

As process mining techniques are maturing and more event data become available, we no longer want to restrict analysis to a single process. We would like compare different variants of the process or different groups of cases. Organizations are interested in *comparative process mining* to understand how processes can be improved. We propose to use *process cubes* as a way to organize event data in a *multi-dimensional data structure tailored towards process mining*.

To illustrate the process cube concept, we used a data set containing partial information about the study behavior of students. When do they view video lectures? What is the effect of viewing these lectures on the exam results? Learning analytics is the broader field that aims to answer such questions. It is defined as "the gathering and analysis of and reporting on data relating to students and their environment for the purpose of gaining a better understanding of and improving education and the environment in which it is provided" [40]. The terms Learning Analytics (LA) and Educational Data Mining (EDM) are used interchangeably and both advocate the intelligent use of event data [17]. Although Pechenizkiy et al. explored the use of process mining in the context of LA and EDM, most of the existing analysis approaches are not process-centric [37, 43, 17]. We consider the comparison of the learning processes inside and between courses as an essential ingredient for a better understanding of the study behavior of students.

The results presented in this paper are only the starting point for a more comprehensive analysis of learning behavior. First of all, we would like to address the foundational challenges described in Section 5 (i.e., refining the process cube concept and improving performance in higher-dimensional cubes). Currently, major implementation efforts are ongoing to provide better support for process cubes. We are developing a solution embedded in ProM (work of Shengnan Guo) and a solution based on database technology calling ProM plug-ins from outside ProM (work of Alfredo Bolt). The latter solution is related to calling ProM plug-ins from other tools like RapidMiner [32] and KNIME. Next, we would like to apply the process cube notion to all video lectures recorded at Eindhoven University of Technology and offer such an in-depth analysis as a service for teachers and students. Finally, we would like to apply comparative

process mining based on process cubes to new data sets. For example, we will use process cubes to analyze the Massive Open Online Course (MOOC) on "Process Mining: Data science in Action" [4]. Moreover, we envision that more and more data on learning behavior will become available from very different sources. This supports our quest to store event data in such a way that interactive analysis becomes possible. Of course there are also serious privacy concerns. Students should be aware of what is recorded and be able to use it to their advantage. For teachers there is often no need to know the progress of individuals. Often it is sufficient to understand the effectiveness of lectures and teaching material at a group level. Only when students give their consent, one should use the information at the level of individual students.

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 Petri Nets for Process Mining: A Generic Approach. *Distributed and Parallel Databases*, 31(4):471–507, 2013.
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, and J. Liu, editors, *Asia Pacific Conference on Business Process Management (AP-BPM 2013)*, volume 159 of *Lecture Notes in Business Information Processing*, pages 1–22. Springer-Verlag, Berlin, 2013.
4. W.M.P. van der Aalst. Process Mining: Data science in Action. Coursera Course, November 2014. https://www.coursera.org/course/procmin.
5. 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.
6. W.M.P. van der Aalst and S. Dustdar. Process Mining Put into Context. *IEEE Internet Computing*, 16(1):82–86, 2012.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.

12. 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.
13. 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.
14. 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.
15. S. Brasch, G. Fuellen, and L. Linsen. VENLO: Interactive Visual Exploration of Aligned Biological Networks and Their Evolution. In L. Linsen, H. Hagen, B. Hamann, and H. Hege, editors, *Visualization in Medicine and Life Sciences II*, Mathematics and visualization, pages 231–249. Springer-Verlag, Berlin, 2012.
16. 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.
17. T. Calders and M. Pechenizkiy. Introduction to the Special Section on Educational Data Mining. *SIGKDD Explorations Newsletter*, 13(2):3–6, 2012.
18. 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.
19. 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.
20. S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technology. *ACM Sigmod Record*, 26(1):65–74, 1997.
21. 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.
22. 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.
23. 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.
24. 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.
25. P. Gorissen. *Facilitating the Use of Recorded Lectures: Analysing Students' Interactions to Understand Their Navigational Needs.* Phd thesis, Eindhoven University of Technology, June 2013.
26. 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.
27. IEEE Task Force on Process Mining. Process Mining Manifesto. In *BPM Workshops*, volume 99 of *Lecture Notes in Business Information Processing*. Springer-Verlag, Berlin, 2011.

28. IEEE Task Force on Process Mining. XES Standard Definition. www.xes-standard.org, 2013.
29. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering Block-structured Process Models from Event Logs: A Constructive Approach. In J.M. Colom and J. Desel, editors, *Applications and Theory of Petri Nets 2013*, volume 7927 of *Lecture Notes in Computer Science*, pages 311–329. Springer-Verlag, Berlin, 2013.
30. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In N. Lohmann, M. Song, and P. Wohed, editors, *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2014)*, volume 171 of *Lecture Notes in Business Information Processing*, pages 66–78. Springer-Verlag, Berlin, 2014.
31. T. Mamaliga. Realizing a Process Cube Allowing for the Comparison of Event Data. Master's thesis, Eindhoven University of Technology, Eindhoven, 2013.
32. R. Mans, W.M.P. van der Aalst, and E. Verbeek. Supporting Process Mining Workflows with RapidProM. In L. Limonad and B. Weber, editors, *Business Process Management Demo Sessions (BPMD 2014)*, volume 1295 of *CEUR Workshop Proceedings*, pages 56–60. CEUR-WS.org, 2014.
33. 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.
34. 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.
35. 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.
36. I. Naydenova and K. Kaloyanova. Sparsity Handling and Data Explosion in OLAP Systems. In *Mediterranean Conference on Information Systems (MCIS 2010)*, page 62. AIS Electronic Library, 2010.
37. M. Pechenizkiy, N. Trcka, E. Vasilyeva, W.M.P. van der Aalst, and P. De Bra. Process Mining Online Assessment Data. In C. Romero, S. Ventura, M. Pechenizkiy, and R. Baker, editors, *Educational Data Mining (EDM 2009)*, pages 279–288. www.educationaldatamining.org, 2009.
38. 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.
39. 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.
40. G. Siemens and R. Baker. Learning Analytics and Educational Data Mining: Towards Communication and Collaboration. In *Proceedings of the International Conference on Learning Analytics and Knowledge (LAK '12)*, pages 252–254, New York, USA, 2012. ACM.
41. 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.

42. 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.
43. N. Trcka, M. Pechenizkiy, and W.M.P. van der Aalst. Chapter 9: Process Mining from Educational Data. In C. Romero, S. Ventura, M. Pechenizkiy, and R. Baker, editors, *Handbook of Educational Data Mining*, Data Mining and Knowledge Discovery Series, pages 123–142. Taylor and Francis, 2010.
44. 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.
45. 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.
46. 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.