A General Framework for Correlating Business Process Characteristics

Massimiliano de Leoni[1,2⋆], Wil M.P. van der Aalst[2], and Marcus Dees[3]

[1] University of Padua, Padua, Italy
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
[3] Uitvoeringsinstituut Werknemersverzekeringen (UWV), The Netherlands
{m.d.leoni, w.m.p.v.d.aalst}@tue.nl,marcus.dees@uwv.nl

Abstract. Process discovery techniques make it possible to automatically derive process models from event data. However, often one is not only interested in discovering the control-flow but also in answering questions like "What do the cases that are late have in common?", "What characterizes the workers that skip this check activity?", and "Do people work faster if they have more work?", etc. Such questions can be answered by combining process mining with classification (e.g., decision tree analysis). Several authors have proposed ad-hoc solutions for specific questions, e.g., there is work on predicting the remaining processing time and recommending activities to minimize particular risks. However, as shown in this paper, it is possible to unify these ideas and provide a general framework for deriving and correlating process characteristics. First, we show how the desired process characteristics can be derived and linked to events. Then, we show that we can derive the selected dependent characteristic from a set of independent characteristics for a selected set of events. This can be done for any process characteristic one can think of. The approach is highly generic and implemented as plug-in for the *ProM* framework. Its applicability is demonstrated by using it to answer to a wide range of questions put forward by the UWV (the Dutch Employee Insurance Agency).

1 Introduction

The interest in process mining is fueled by the rapid growth of event data available for analysis. Moreover, there is increasing pressure to make Business Process Management (BPM) more "evidence based", i.e., process improvements and innovations are more and more driven by facts. Process mining often starts with *process discovery*, i.e., automatically learning process models based on raw event data. Once there is a process model (discovered or made by hand), the events can be replayed on the model to *check conformance* and to *uncover bottlenecks* in the process. However, such analyses are often only the starting point for providing initial insights. When discovering a bottleneck or frequent deviation, one would like to understand why it exists. This requires the correlation of different *process characteristics*. These characteristics can be based on the control-flow (e.g., the next activity going to be performed), the data-flow (e.g., the amount of money involved), the time perspective (e.g., the activity duration or the remaining time to the end of the process), the organization perspective (e.g., the resource

⋆ The work of Dr. de Leoni is supported by the Eurostars - Eureka project PROMPT (E!6696).

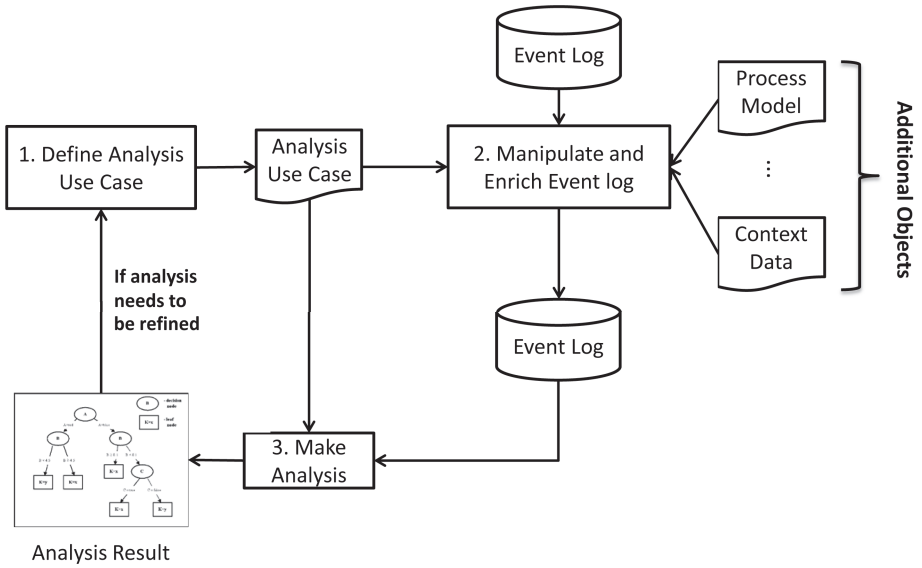

Fig. 1. The general framework proposed in this paper: based on an analysis use case the event log is preprocessed and used as input for classification. Based on the analysis result, the use case can be adapted to gather additional insights.

going to perform a particular activity), or, in case a normative process model exists, the conformance perspective (e.g., the skipping of a mandatory activity).

The study of these characteristics and how they influence each other is of crucial importance when an organization aims to improve and redesign its own processes. Many authors have proposed techniques to relate specific characteristics in an ad-hoc manner. For example, several approaches have been proposed to predict the remaining processing time of a case depending on characteristics of the partial trace executed [1–3]. Other approaches are only targeted to correlating certain predefined characteristics to the process outcome [4–6] or the violations of business rules [7].

These problems are specific instances of a more general problem, which is concerned with *relating any process or event characteristic to other characteristics associated with single events or the entire process*. This paper proposes a framework to solve the more general correlation problem and provides a very powerful tool that unifies the ad-hoc approaches described in literature. This is achieved by providing *(1)* a broad and extendable set of characteristics related to time, routing, ordering, resource allocation, workload, and deviations, and *(2)* a generic framework where any characteristic (dependent variable) can be explained in term of correlations with any set of other characteristics (independent variables), For instance, the involvement of a particular resource or routing decision can be related to the elapsed time, but also the other way around: the elapsed time can be related to resource behavior or routing.

Figure 1 illustrates the framework proposed in this paper. Starting point is an *event log*. For each process instance (i.e., case) there is a trace, i.e., a sequence of events.

Events have different *characteristics*. Mandatory characteristics are *activity* and *timestamp*. Other standard characteristics are the *resource* used to perform the activity, *transactional* information (start, complete, suspend, resume, etc.), and *costs*. However, any other characteristics can be associated to an activity (e.g., the age of a patient or size of an order). Characteristics are attached to events as name-value pairs: $(name\_of\_characteristic, value)$. Event characteristics can be also concerned with the *context* of the event, case, process, or organization. The process context is acknowledged to be very important to find correlations with process and event characteristics [8–10].

For instance, it is possible to add case-related contextual information, such as the remaining flow time or the elapsed time since the process instance started. Also properties of the resource executing the event (e.g., workload of the resource) can be added. We can also add the next activity as a characteristic of an event. One can even add conformance checking results and external context such as weather information to events as characteristics. The ultimate goal of our framework is to mine decision trees that explain the value of one characteristic, the *dependent characteristic*, in terms of the other characteristics, the *independent characteristics*.

In addition to decision trees, many other machine-learning techniques exist and some have already been applied in BPM, such as Bayesian Networks [11], Case-Based Reasoning [12] and Markov Models [3]. These are certainly valuable but they are only able to make correlations for single instances of interest or to return significant examples of relevant instances. Conversely, we aim to aggregate knowledge extracted from the event logs and return it as decision rules. Association rules [8] could have been an alternative, but decision trees have the advantage of clearly highlighting the characteristics that are most discriminating. Regression analysis [13] would be only applicable to find numerical correlations and, hence, it could not be employed if the dependent characteristic is nominal or boolean.

The approach is fully supported by a new package that has been added to the open-source process mining framework *ProM*.[4] The evaluation of our approach is based on a case study involving UWV, a Dutch governmental institute in charge of supplying benefits. In particular, we have employed the approach to answer process-related questions that were relevant for the institution. The results were extremely positive: we could answer the UWV's questions regarding the causes of observed problems (e.g., reclamations of customers). For some problems, we could show surprising root causes. For other problems, we could only show that some suspected correlations were not present.

Section 2 presents the framework and highlights that several well-studied problems are specific instances of the more general problem considered in this paper. Section 3 shows the application of our framework and implementation in the context of UWV. Finally, Section 4 concludes the paper.

2 The Framework

The main input of our framework is an event log.

Definition 1 (Events, Traces and Log). *Let $\mathcal{C}$ and $\mathcal{U}$ be the universe of characteristics and the universe of possible values respectively. An* event e *is an assignment of values*

[4] The *FeaturePrediction* package, see http://www.promtools.org.

to characteristics, i.e. $e \in \mathcal{C} \rightarrow \mathcal{U}$*. In the remainder* $\mathcal{E} = \mathcal{C} \rightarrow \mathcal{U}$ *is the universe of events. A* trace $T \in \mathcal{E}^*$ *is a sequence of events. Let* $\mathcal{T} = \mathcal{E}^*$ *be the universe of traces. An* event log L *is a multi-set of traces, i.e.* $L \in \mathbb{B}(\mathcal{T})$

For each characteristic $c \in \mathcal{C}$, $type(c) \subseteq \mathcal{U}$ denotes the set of possible values. We use a special value $\perp$ for any characteristic c which an event e is not assigning a value to, i.e. $e(c) = \perp$ if $c \notin dom(e)$. Typically, an event refers to an activity that is performed within a certain case by a resource at a given timestamp. In our framework, these are merely treated as any event characteristics: *Activity*, *Case*, *Resource*, *Timestamp*, respectively. The occurrence of an event, i.e. the execution of an activity, can assign new values to any subset of characteristics.

Our framework aims to support so-called *analysis use cases*.

Definition 2 (Analysis Use Case). *An* analysis use case *is a triple* $(\mathcal{C}_d, c_r, F)$ *consisting of*

- *a dependent characteristic* $c_r \in \mathcal{C} \setminus \mathcal{C}_d$*,*
- *a set* $\mathcal{C}_d \subset \mathcal{C}$ *of independent characteristics,*
- *an event-selection filter* $F \subseteq \mathcal{E}$*, which characterizes the events that are retained for the analysis.*

The output of an analysis use case is a decision tree. Decision trees classify instances (in our case events) by sorting them down in a tree from the root to some leaf node. Each non-leaf node specifies a test of some attribute (in our case, an independent characteristic) and each branch descending from that node corresponds to a range of possible values for this attribute. Each leaf node is associated to a value of a class attribute (in our case, the dependent characteristic). A path from root to a leaf represents a classification rule. There exist many algorithms to build a decision tree starting from a training set [13]. Our framework is agnostic with respect to specific algorithms used for decision-tree learning. In our implementation we rely on the C4.5 algorithm, which can handle continuous attributes efficiently and is good at pruning the final decision tree [13]. However, any other classification algorithm could have been used. In the remainder, given a set of instances (i.e. events) $\mathcal{I} \in 2^{(\mathcal{C} \rightarrow \mathcal{U})}$, a set $\mathcal{C}_d \subset \mathcal{C}$ of independent variables (i.e., the independent characteristics) and a dependent variable $c_r \in \mathcal{C} \setminus \mathcal{C}_d$, the procedure to train a decision tree is denoted as $\mathit{generateTree}(\mathcal{I}, \mathcal{C}_d, c_r)$.

Algorithm 1 describes our approach to build a decision tree based on an event log and an analysis use case. The input consists of an event log and an analysis use case. At the end, set $\mathcal{I}$ contains all instances that are used to train the decision tree. This set is populated with every event e of the log that is not filtered out by the event-selection filter F.

If the dependent characteristic c_r is defined over a continuous domain, value $e(c_r)$ is discretized before event e is added to the instance set $\mathcal{I}$. Decision trees do not support continuous class variables. Therefore, continuous characteristics need to be discretized to be used as dependent. In the algorithm, the procedure of discretization is abstracted as a function $\mathit{discretize}(val, c, L, n)$ that, given a characteristic c, a value $val \in type(c)$, the set of values observed in an event log L, and a number n of discretization intervals, returns a value over a discrete domain. Literature provides several ways to discretize

Algorithm 1: Generate Decision Tree

Input: Event Log $L \in \mathbb{B}(\mathcal{T})$, An analysis use case $(\mathcal{C}_d, c_r, F)$, the number n of discretization intervals.
Result: Decision Tree

$\mathcal{I} \leftarrow \varnothing$
foreach $T \in L$ **do**
  foreach $e \in L$ **do**
    if $e \in F$ **then**
      if $type(c_r)$ *is continuous* **then**
        $e(c_r) \leftarrow discretize(e(c_r), c_r, L, n)$
      end
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{e\}$
    end
  end
end
$generateTree(\mathcal{I}, \mathcal{C}_d, c_r)$

dependent variables. While our approach can use any discretization technique, our implementation provides two specific ones: *equal-width binning* and *equal-frequency binning* [14]. Given a number n of intervals, the former divides the set of possible values $type(c)$ into n equal-width intervals, assigning a discrete value to each of them. Continuous values are transformed into discrete values according to the intervals they fall into. The *equal-frequency binning* approach tries to transform values more evenly: intervals are of different sizes, choosing them such that (roughly) the same number of observed values falls into each one.

As mentioned before, for many analysis use cases we need dependent or independent characteristics that are not readily available in the event log. Similarly, using business domain knowledge, an analyst may want to verify a reasonable hypothesis of the existence of a correlation to a given set of independent characteristics, which may not be explicitly available in the event log. However, values for many interesting characteristics can easily be derived from the event data in the event log. In some cases we will even derive characteristics from information sources outside the event log (weather information, stock index, etc.).

We provide a powerful framework to manipulate event logs and obtain a new event log that suits the specific analysis use case, e.g. events are enriched with additional characteristics.

Definition 3 (Trace and Log Manipulation). *Let $\mathcal{T}$ be the universe of traces and event logs and let $L \in \mathbb{B}(\mathcal{T})$ be an event log. A* trace manipulation *is a function $\delta_L \in \mathcal{T} \rightarrow \mathcal{T}$.*

In the remainder, given a trace-manipulation function δ_L, we also allow δ_L to be applied to an entire log L, thus returning a new log obtained by applying the trace-manipulation function to all traces in L.

Table 1 shows a taxonomy of trace manipulations, grouping them by the process perspective that they take into account. All manipulations shown have been implemented in ProM; the generality of the framework also makes it easy to add new manipulations. Due to space limitations, we can only discuss some of them:

Table 1. A Taxonomy of Trace Manipulations currently available in the operationalization.

Perspective	**Trace Manipulations**
Control-flow	Number of Executions of Activity a, Next Activity in the Trace, Previous Activity in Trace.
Resource	Workload per Resource, Total Workload.
Time	Time Elapsed since the Start of the Case, Remaining Time Until the End of Case, Activity Duration.
Data-flow	Latest Recorded Value of Characteristic c Before Current Event e, Latest Recorded Value of Characteristic c After Event e, Case-Level Abstraction.
Conformance	Trace Fitness, Number of Not Allowed Executions of Activity a Thus Far (moves on log in alignment), Number of Missing Executions of Activity a Thus Far (moves on model in alignment), Number of Correct Executions of Activity a Thus Far (synchronous moves), Satisfaction of Formula F Considering the Prefix Trace Until Current Event e.

- **Next Activity in the Trace.** It augments each event with an extra attribute that contains the name of the next activity in the trace (or $\perp$ for the last event)
- **Latest Recorded Value of Characteristic c Before Current Event e.** It enriches each event e with the latest value assigned to characteristic c before e in the trace.
- **Latest Recorded Value of Characteristic c After Current Event e.** It enriches each event e with the latest value assigned to characteristic c after e in the trace. It differs from the manipulation *Latest Recorded Value of Characteristic c Before Current Event e* in that the value is taken after the execution of e. If e does not write a value for c, the value before and after e coincides.
- **Case-Level Abstraction.** This replaces all the events the trace with two events, the *case-start* and *case-complete* event. The case-start event is associated with the same values of the characteristics as the first event of the trace. The case-end event is associated with the last recorded values for all characteristics. For both events, the value of the *Activity* characteristic is overwritten with value "Case".
- **Workload per Resource.** It associates each event e with the work-load for the resource that has triggered the event, i.e. the number of activities under execution by $e(Resource)$, at the time the event occurred.
- **Total Workload.** It associates events with the number of activities being executed at the time the event occurred.
- **Activity Duration.** Each (complete) event is associated with a (integer-typed) characteristic that indicates the duration of completing the activity associated with the event.

The last three characteristics in the list require an analysis of the entire log, i.e., the scope is not limited to a single trace. This is the reason why trace-manipulation function $\delta_L(T)$ depends not just on trace T but also L.

Indeed, the values to associate with each event can be derived by replaying the event log and counting the number of activities being executed in each moment in time. It is not necessary that the event log records the starting and completion of each activity. Although that would be preferred as the workload would be calculated exactly, we have implemented algorithms that can estimate the workload by only using the activity completion events. In our implementation, we estimate the start time of activities as

Table 2. Fragment of a hospital's event log with four traces. The gray columns have been added after applying two of the trace manipulations in Table 1: Next Activity in the Trace and Time Elapsed since the Start of the Case. `NextActivityInTrace` and `ElapsedTime` are the names of the characteristics that are added as result of these manipulations.

Case	Timestamp	Activity	Resource	Cost	NextActivityInTrace	ElapsedTime
1	1-12-2011:11.00	Preoperative Screening	Giuseppe	350	Laparoscopic Gastrectomy	0 days
1	2-12-2011:15.00	Laparoscopic Gastrectomy	Simon	500	Nursing	1.16 days
1	2-12-2011:16.00	Nursing	Clare	250	Laparoscopic Gastrectomy	1.20 days
1	3-12-2011:13.00	Laparoscopic Gastrectomy	Paul	500	Nursing	2.08 days
1	3-12-2011:15.00	Nursing	Andrew	250	First Hospital Admission	2.16 days
1	4-12-2011:9.00	First Hospital Admission	Victor	90	$\perp$	3.92 days
2	7-12-2011:10.00	First Hospital Admission	Jane	90	Laparoscopic Gastrectomy	0 days
2	8-12-2011:13.00	Laparoscopic Gastrectomy	Giulia	500	Nursing	1.08 days
2	9-12-2011:16.00	Nursing	Paul	250	$\perp$	2.16
3	6-12-2011:14.00	First Hospital Admission	Gianluca	90	Preoperative Screening	0 days
3	8-12-2011:13.00	Preoperative Screening	Robert	350	Preoperative Screening	1.96 days
3	10-12-2011:16.00	Preoperative Screening	Giuseppe	350	Laparoscopic Gastrectomy	4.08 days
3	13-12-2011:11.00	Laparoscopic Gastrectomy	Simon	500	First Hospital Admission	6.88 days
3	13-12-2011:16.00	First Hospital Admission	Jane	90	$\perp$	7.02 days
4	7-12-2011:15.00	First Hospital Admission	Carol	90	Preoperative Screening	0 days
4	9-12-2011:7.00	Preoperative Screening	Susanne	350	Laparoscopic Gastrectomy	0.66 days
4	13-12-2011:11.00	Laparoscopic Gastrectomy	Simon	500	Nursing	5.84 days
4	13-12-2011:13.00	Nursing	Clare	250	Nursing	5.92 days
4	13-12-2011:19.00	Nursing	Vivianne	250	$\perp$	6.16 days

Table 3. The results after applying the Case-Level Manipulation to the event log shown in Table 2.

Case	Timestamp	Activity	Resource	Cost	NextActivityInTrace	ElapsedTime
1	1-12-2011:11.00	Case	Giuseppe	350	Laparoscopic Gastrectomy	0 days
1	4-12-2011:9.00	Case	Victor	90	$\perp$	3.92 days
2	7-12-2011:10.00	Case	Jane	90	Laparoscopic Gastrectomy	0 days
2	9-12-2011:16.00	Case	Paul	250	$\perp$	2.16 days
3	6-12-2011:14.00	Case	Gianluca	90	Preoperative Screening	0 days
3	13-12-2011:16.00	Case	Jane	90	$\perp$	7.02 days
4	7-12-2011:15.00	Case	Carol	90	Preoperative Screening	0 days
4	13-12-2011:19.00	Case	Vivianne	250	$\perp$	6.16 days

proposed in [15]: assuming no waiting time, the start time of an activity is the latest between the time of completion of the previous activity within the same process instance and the time of completion of the previous activity by the same resource (possibly in a different process instance).

Tables 2 and 3 illustrate the application of some of the manipulation functions of Table 1 to a fragment of an event log. In general, when multiple trace-manipulation functions are applied, the order of application may be important. Table 3 shows the result of the application of the case-level abstraction manipulation after applying *Next Activity in the Trace* and *Time Elapsed since the Start of the Case*. If case-level abstraction was applied before the other two, the final result would be different: characteristic *Next Activity in the Trace* would be given either value "Case" or $\perp$.

Table 1 also illustrates a number of trace manipulations that require additional sources/inputs, such as a process model, declarative or procedural, or a (temporal) logical formula F:

– **Trace Fitness.** Given a process model, it augments each event with a continuous value between 0 and 1, denoting the level of the fitness of the model and the trace to

which the event belongs. Values 1 and 0 denote perfect and extremely poor fitness, respectively.

- **Number of Not Allowed Executions of Activity a Thus Far, Number of Missing Executions of Activity a Thus Far, and Number of Correct Executions of Activity a Thus Far.** These manipulations augment each event with an integer characteristic that denotes the number of "moves on logs" (occurs in reality but disallowed according to the model), "moves on model" (should have occurred according to the model but did not), and "synchronous moves" (model and reality agree) respectively in the prefix until the current event.
- **Satisfaction of Formula F Considering the Prefix Trace Until Current Event e.** It augments each event with a boolean value that states whether a given formula F was satisfied after the event occurred.

The third manipulation in the above list builds on the ProM operationalization of the technique described in [16]. Here Linear Temporal Logic (LTL) is used to specify F. The others rely on the ProM implementation of the techniques discussed in [17, 18], which are concerned with finding an alignment of traces in the log with, respectively, procedural and declarative process models.

Tables 4 and 5 show how six examples of correlation problems can be formulated as analysis use cases. In the tables, the original log denotes the log before any trace manipulation. Trace manipulations are applied in the exact order as they are enumerated in the list. As the examples show, prediction problems are, in fact, correlation problems. When a correlation is observed in the past, one can predict that the same correlation is going to be observed for future process instances, as well.

For most of the problems shown in Tables 4 and 5, research work has already been conducted, yielding ad-hoc solutions. Our framework attempts to solve the more general problem, i.e., finding any type of correlation among arbitrary process characteristics at any level (event, case, process, resource, etc.). By solving the more general problems, we can support existing analyses but also many more.

Some of the analysis use cases in Tables 4 and 5 have been used as intermediate results to solve other problems. For instance, Ghattas et al. [4] uses the answer to Problem #2 as information to drive how to redesign the process to improve the process' outcomes. Similarly, the solutions of Problems #1 and #4 are used in [7] and [6], respectively, as input to provide a run-time support to suggest the next activities to work on.

3 Evaluation with a real-life case study

This section illustrates how our framework can be used to help UWV. UWV (Employee Insurance Agency) is an autonomous administrative authority to implement employee insurances and provide labor market and data services. One of the core tasks of UWV is ensuring that benefits are provided quickly and correctly when a Dutch resident, hereafter customer, cannot immediately find a new job after ceasing the previous. UWV is facing various undesired process executions and is interested in discovering the root-causes of a variety of problems identified by UWV's management. In these analysis use cases, we are looking at the process to deal with requests of unemployment benefits. An instance of this process starts when a customer applies. Subsequently, checks are

Table 4. Five example analysis use cases illustrating the generic nature of the framework presented.

Problem #1: Run-time predictions of violations of formula *F*.
Description: The aim is to predict, given the current status of the process instances, the next activities to work on to maximize the chances of achieving certain business goals expressed as formula F. In [7], an ad-hoc solution is proposed for this problem where formulas are expressed in LTL.
Dependent Characteristic: Satisfaction of Formula F Considering the Prefix Trace Until Current Event.
Independent Characteristics: For each characteristic c of the original event log, the Latest Recorded Value of c Before the Current Event; Activity Name; the resource name.
Event Filter: Every event is retained.
Trace Manipulation: Satisfaction of Formula F Considering the Prefix Trace Until Current Event; for each characteristic c of the original event log, the Latest Recorded Value of c Before the Current Event.

Problem #2: Prediction of the outcomes of the executions of process instances.
Description: The aim is to predict the outcome of a case. Predictions are computed using a set of complete process instances that are recorded in the event log. The last event of each trace is associated with a characteristic *Outcome* to which it is assigned a numeric value that indicates the quality of the outcome. The prediction is done at case level: one instance for learning is created for each trace in the event log. The outcome of the entire trace is predicted rather than of single activities. In [4], an ad-hoc solution is proposed for this problem.
Dependent Characteristic: *Outcome*
Independent Characteristics: Each characteristic c of the original event log, except *Outcome*.
Event Filter: Every case-complete event is retained.
Trace Manipulation: Case-level Abstraction.

Problem #3: Mining of decisions that determine the activity to execute after the execution of an activity a.
Description: The purpose is to predict the conditions that discriminate which activity is executed after a given activity a. Predictions are computed using a set of complete process instances that are recorded in the event log. In particular, only the events referring to activity a are used. In [19], an ad-hoc solution is proposed for this problem.
Dependent Characteristic: Next Activity In the Trace.
Independent Characteristics: For each characteristic c of the original event log, the Latest Recorded Value of c After the Current Event.
Event Filter: Every event e for activity a is retained, i.e. every event e such that $e(Activity) = a$.
Trace Manipulation: For each characteristic c of the original event log, the Latest Recorded Value of c After the Current Event; Next Activity In the Trace.

Problem #4: Prediction of faults during business process executions.
Description: The purpose is to predict whether or not a running instance is going to complete with a fault. If completed with a fault, its magnitude is also predicted. Predictions are computed using a set of complete process instances that are recorded in the event log. If a fault has occurred for a given completed instance, the first event of the corresponding trace is associated with a characteristic *Fault* to which a value is assigned that indicates the magnitude. If no fault is occurred, the first event is associated with a characteristic *Fault* to which a value 0 is assigned. In [6], an ad-hoc solution is proposed for this problem.
Dependent Characteristic: The value of *Fault* after the Current Event.
Independent Characteristics: For each characteristic c of the original event log besides *Fault*, the Latest Recorded Value of c After the Current Event; for each activity a, the Number of Executions of a; Elapsed Time Since the Start of the Case; Activity Name; Resource Name.
Event Filter: Every event is retained.
Trace Manipulation: For each characteristic c of the original event log, the Latest Recorded Value of c After the Current Event; for each activity a, the Number of Executions of Activity a; the Elapsed Time since the Start of the Case.

Problem #5: Prediction of the executor of a certain activity a.
Description: The purpose is to mine the conditions that determine which resource is going to work on a given activity a at a certain moment during the process execution.
Dependent Characteristic: *Resource Name*
Independent Characteristics: Potentially, any characteristic of the original event log as well as any characteristic with which events can be augmented. Every characteristic can be relevant for this prediction.
Event Filter: Every event for activity a is retained, i.e. every event e such that $e(Activity) = a$.
Trace Manipulation: Depending on the scenario, any manipulation but Case-Level abstraction can be relevant.

Table 5. An additional analysis use case illustrating the generic nature of the framework presented.

Problem #6: Prediction of the Remaining Time to the end of process instances.
Description: The purpose is to predict the remaining time until the end of process instances on the basis of the current state, which consists of the number of executions of each process activity and the current values of process variables. It is similar to [2] when a multi-set abstraction is used, with, additionally, the current values of process variables are also taken into account.
Dependent Characteristic: Remaining Time Until The End of the Case.
Independent Characteristics: For each process activity a, the number of executions of activity a; for each characteristic c of the original event log, the Latest Recorded Value of c After the Current Event.
Event Filter: Every event is retained.
Trace Manipulation: For each characteristic c of the original event log, the Latest Recorded Value of c After the Current Event; for each process activity a, the Number of Executions of Activity a; the Remaining Time Until the End of the Case.

performed to verify the entitlement conditions. If the checks are positive, the instance is being executed for the entire period in which the customer receives the monetary benefits, which are paid in monthly installments. Entitled customers receive as many monthly installments as the number of years for which they were working. Therefore, an instance can potentially be executed for more than one year. During the entire period, customers must comply with certain duties, otherwise a customer is sanctioned and a reclamation is opened. When a reclamation occurs, this directly impacts the customer, who will receive lower benefits than expected or has to return part of the benefits. It also has negative impact from UWV's viewpoint, as this tends to consume lots of resources and time. Therefore, UWV is interested to know the root causes of opening reclamations to reduce their number. If the root causes are known, UWV can predict when a reclamation is likely going to be opened and, hence, it can enact appropriate actions to prevent it beforehand. In order to discover the root causes, UWV formulated four questions:

Q1 Are customer characteristics linked to the occurrence of reclamations? And if so, which characteristics are most prominent?
Q2 Are characteristics concerned with how process instances are executed linked to the occurrence of reclamations? And if any, which characteristics matter most?
Q3 If the prescribed process flow is not followed, will this influence whether or not a reclamation occurs?
Q4 When an instance of the unemployment-benefit payment process is being handled, is there any characteristic that may trigger whether a reclamation is going to occur?

Table 6 enumerates some of the analysis use cases that have been performed to answer the questions above. The analyses have been performed using a UWV's event log containing 2232 process instances and 77551 events. Since the original event log contains more than 100 characteristics, it is not possible to punctually detail single characteristics that have been included or excluded from the analyses. The remainder of this section details how the analysis use cases have been used to answer the four questions above.

Question Q1. To answer this question, we performed the use case **U1** in Table 6. The results of performing this analysis are represented through the decision tree in Figure 2. In particular, the screenshot refers to our implementation in ProM. The implementation

Table 6. Some of the analysis use cases analyzed to provide an answer to the correlation problems raised by UWV.

U1. Are customer characteristics linked to the occurrence of reclamations?
Description: We aim to correlate the number of executions of activity *Reclamation* to the customer characteristics. We are interested in all decision-tree paths that lead to a number of executions of activity Reclamation greater than 0.
Dependent Characteristic: Number of Executions of Activity *Reclamation*.
Independent Characteristics: All characteristics of the events in the original log that refer to customers properties.
Event Filter: Every case-complete event is retained.
Trace Manipulation: Number of executions of Activity *Reclamation*; Case-Level Abstraction

U2. Are characteristics concerned with how process instances are executed linked to the occurrence of reclamations? – Iteration 1
Description: We aim to correlate the number of execution of activity *Reclamation* to process characteristics, the number of executions of all activities and the elapsed time, i.e., the time to complete a process instance. We are interested in all decision-tree paths that lead to a number of executions of activity Reclamation greater than 0.
Dependent Characteristic: Number of Executions of Activity *Reclamation*.
Independent Characteristics: For each process activity a besides *Reclamation*, Number of Executions of a; Time Elapsed Since the Start of the Case; all characteristics of the events of the original log that refer to the outcomes of process instances; Timestamp.
Event Filter: Every case-complete event is retained.
Trace Manipulation: For each process activity a, Number of Executions of a; Time Elapsed Since the Start of the Case; Case-Level Abstraction.

U3. Are characteristics concerned with how process instances are executed linked to the occurrence of reclamations? – Iteration 9
Description: We aim to correlate the number of execution of activity *Reclamation* to process characteristics and the number of executions of most of activities. We are interested in all decision-tree paths that lead to a number of executions of activity Reclamation greater than 0.
Dependent Characteristic: Number of executions of activity *Reclamation*.
Independent Characteristics: For each process activity a besides *Reclamation* and *Call Contact door HH deskundige*, Number of Executions of a; All characteristics of the events in the original log that refer to the outcomes of process instances, besides *Soort Vaststelling* and 49 more.
Event Filter: Every case-complete event is retained.
Trace Manipulation: For each process activity a, Number of Executions of a; Case-Level Abstraction.

U4. If the prescribed process flow is not followed, will this influence whether or not a reclamation occurs?
Description: We aim to correlate the number of execution of activity *Reclamation* to process characteristics, the number of executions of most of activities as well as to the deviations wrt. the prescribed process model. We are interested in all decision-tree paths that lead to a number of executions of activity Reclamation greater than 0.
Dependent Characteristic: Number of executions of activity *Reclamation*.
Independent Characteristics: Trace Fitness; for each process activity a besides *Reclamation*, the Number of Not-Allowed Executions of a, the Number of Missing Executions of a and the Number of Correct Executions of a; Number of Executions of Activity *Reclamation*.
Event Filter: Every event is retained.
Trace Manipulation: Trace Fitness; for each process activity a, the Number of Not-Allowed Executions of a Thus Far, the Number of Missing Executions of a Thus Far and the Number of Correct Executions of a Thus Far; Number of Executions of Activity *Reclamation*.

U5. When an instance of the unemployment-benefit payment process is handled, is there any characteristic that may trigger whether a reclamation is going to occur?
Description: We aim to predict when a reclamation is going to follow any process activity. For this purpose, we predict which activity is going to follow any process activity and, then, we focus on those paths leading to predicting reclamation as next activity in trace.
Dependent Characteristic: Next Activity in Trace.
Independent Characteristics: For each process activity a besides *Call Contact door HH deskundige*, the Number of Executions of a; All characteristics of the events in the original log that refer to the outcomes of process instances
Event Filter: Every event is retained.
Trace Manipulation: For each process activity a, Number of Executions of a; Next Activity in the Trace.

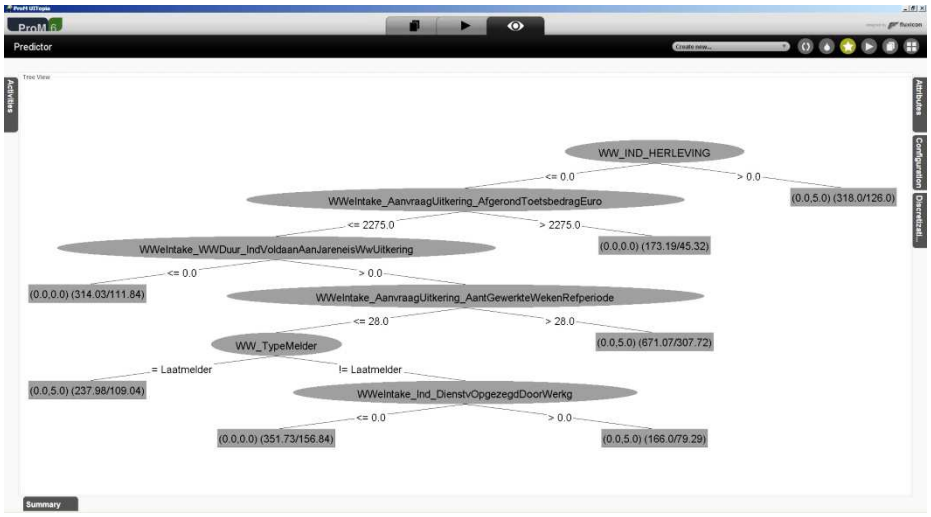

Fig. 2. A screenshot of the framework's implementation in ProM that shows the decision tree used to answer question Q1.

allows the end user to configure a number of parameters, such as the level of decision-tree pruning, the minimum number of instances per leaf or the discretization method. In this way, the user can try several configurations, thus, e.g., balancing between over- and under-fitting. In particular, the screenshot refers to the configuration in which the minimum number of instances per leaf is set to 100 and the number of executions of *Reclamation* is discretized as two values: `(0.0,0.0)` and `(0.0,5.0)`. When the number of executions of *Reclamation* is 0, this is shown as `(0.0,0.0)`; conversely, any value greater than 0 for the number of executions is discretized as `(0.0,5.0)`. The use cases *U2*, *U3*, *U4* also use the number of executions of *Reclamation* as dependent characteristic. We used the same discretization for those use cases, as well.

Looking at the tree in Figure 2, some business rules seem to be derived. For instance, if the customer is a recurrent customer ($WW\_IND\_HERLEVING > 0$), a reclamation occurs, i.e. the leaf is labelled as `(0.0,5.0)`.[5] If this correlation really held, it would be quite unexpected: recurrent customers tend to disregard their duties. Nonetheless, the label is also annotated with `318.0/126.0`, which indicates that a reclamation is not opened for 126 out of the 318 recurrent customers (39%). Though not very strong, a correlation seems to exist between being recurrent customers and incurring in reclamations. Further investigation is certainly needed; perhaps, additional customer's characteristics might be needed to better discriminate but they are currently not present in the event log used for analysis.

Question Q2. Firstly, we performed the analysis use case **U2**. We obtained a decision tree that showed correlations between the number of reclamations and certain charac-

[5] Customers are recurrent if they apply for monetary benefits multiple times because they find multiple temporary jobs and, hence, they become unemployed multiple times.

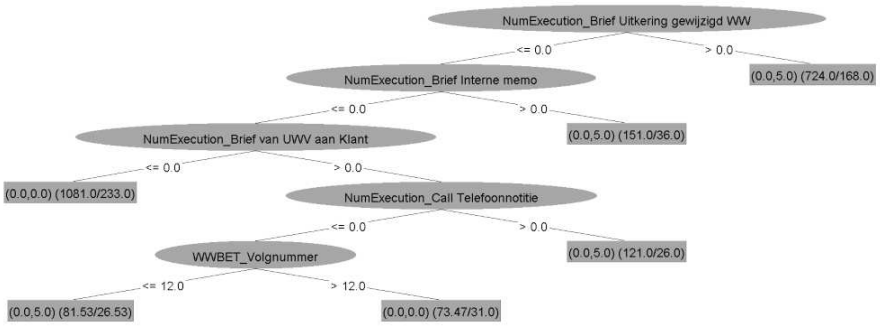

Fig. 3. The decision tree used to answer question Q2.

teristics that are judged as trivial by UWV. For instance, there was a correlation of the number of reclamations with *(1.)* the method of payment of the benefit installments to customers and *(2.)* the number of executions of activity *Call Contact door HH deskundige*, which is executed to push customers to perform their duties. Being these correlations considered trivial by UWV, the respective characteristics should be left out of the analysis. So, we excluded these characteristics from the set of independent characteristics and repeated the analysis. We refined the use case analysis multiple times by removing more and more independent characteristics. After 9 iterations, we performed an analysis use case that led to satisfactory results. This use case is denoted as **U3** in Table 6. The results of performing this analysis are represented through the decision tree in Figure 3, which classifies 77% of the instances correctly.

This tree illustrates interesting correlation rules. Reclamations are usually not opened in those process instances in which *(1.)* UWV never informs (or has to inform) a customer about changes in his/her benefits (the number of executions of *Brief Uitkering gewijzigd WW* is 0), *(2.)* UWV's employees do not hand over work to each other (the number of executions of *Brief Interne Memo* is 0) and *(3.)* either of the following conditions holds:

- No letter is sent to the customers (the number of executions of *Brief van UWV aan Klant* is 0);
- At least one letter is sent but UWV never calls the customer (the number of executions of *Call Telefoonnotitie* is equal to 0) and, also, the number of months for which the customer is entitled to receive a benefit is more than 12.

From this analysis, we can conclude that UWV should reduce the hand-over of work. Moreover, it should pay more attention to customers when their situation changes, e.g. they find a new job. When customers find a job, they start having a monetary income, again. The problem seems to be related to the customers who often do not provide information about the new job on time. In these cases, their benefits are not stopped or reduced when they should. Consequently, a reclamation needs to be opened because these customers need to return the amount that was overpaid to them. Conversely, if a customer has already received benefits for 12 months, it is unlikely that a reclamation

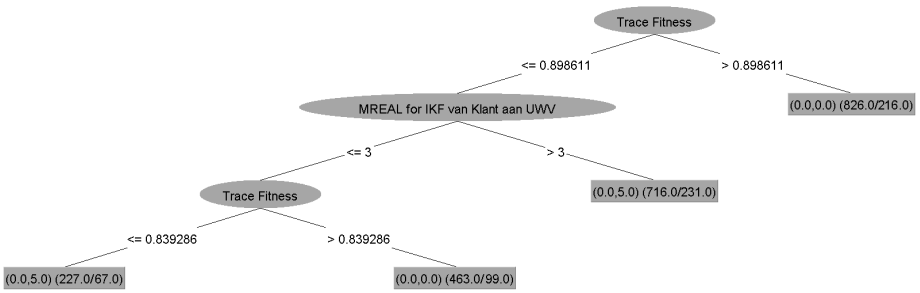

Fig. 4. The decision tree used to answer question Q3.

is going to occur. This can be motivated quite clearly and it is again related to the presence of changes of the customer's job situation. If benefits are received for more than 12 months, the customer has not found a job in latest 12 months and, thus, it is probably going to be hard for him to find one. So, UWV does not have to pay much attention to customers entitled to long benefits when they aim to limit the number of reclamations.

Question Q3. The answer to this question is given by performing the analysis use case **U4**. This use case relies on a process model that describes the normal execution flow. This model was designed by hand, using knowledge of the UWV domain. The results of performing **U4** is represented by the decision tree in Figure 4. Trace Fitness is measured as a value between 0 and 1 (see [17]). Values 1 and 0, respectively, denote perfect and poor fitness between the expected behaviour, represented by the process model, and the actual behaviour, which is recorded in the event log. Analyzing the decision tree, a correlation is clear between trace fitness and the number of reclamations. 610 out of the 826 process executions (nearly 70%) with fitness higher than 0.89 do not comprise any reclamation. Therefore, it seems crucial for UWV to make the best to follow the normal flow, although this is often made difficult by a hasty behavior of customers. This rule seems quite reliable and is also confirmed by the fact that 70% of the executions with fitness lower than 0.83 incur in reclamations.

The decision tree contains an intermediate node labelled *MREAL for IKF van Klant aan UWV*. This characteristic refers to the number of missing executions of activity *IKF van Klant aan UWV*. This activity is executed in a process instance every time that UWV receives a declaration form from the customer. UWV requests customers to send a form every month to declare whether or not their condition has changed in the last month, e.g. they found a job. The decision tree states that, when an execution deviates moderately, i.e. the fitness is roughly between 0.83 and 0.89, a reclamation is still unlikely being opened if the customer forgets to send the declaration form for less than 3 months (not necessarily in a row). Please note that, since traces are quite long, considering how fitness is computed, a difference of 0.06 in fitness can be quite remarkable. This rule is quite reliable since it holds in 79% of cases. Therefore, it is

worthwhile for UWV to enact appropriate actions (such as calling by phone) to increase the chances that customers send the declaration form every month.

Question Q4. The answer to this question is given by performing the analysis use case **U5**. We built a decision tree for this use case by limiting the minimal number of instances per leaf to 50. We are interested in tree paths that lead to *Reclamation* as next activity in the trace. Unfortunately, the F-measure for *Reclamation* was very low (0.349), which indicates that *it is not possible to reliably estimate if a reclamation is going to occur at a certain moment of the execution of a process instance*. We also tried to reduce the limit of the minimum number of instances per leaf. Unfortunately, the resulting decision tree was not valuable since it overfitted the instance sets: the majority of the leaves were associated to less than 1% of the total number of instances. Conversely, the decision tree with 50 as minimum number of instance per leaf could be useful to predict when a payment is sent out to a customer: the F score for the payment activity is nearly 0.735. Unfortunately, finding this correlation does not answer question Q4.

4 Conclusion

Process mining is not just about discovering the control-flow or diagnosing deviations. It is crucial that certain phenomena can be explained, e.g., "Why are these cases delayed at this point?", "Why do these deviations take place?", "What kind of cases are more costly due to following this undesirable route?", and "Why is the distribution of work so unbalanced". Although numerous analysis approaches have been proposed for specific questions, a generic framework for correlating business process characteristics was missing. In this paper, we presented such a framework and its implementation in ProM. By defining an analysis use case composed of three elements (one dependent characteristic, multiple independent characteristics and a filter), we can create a classification problem. The resulting decision tree aims to describe the dependent characteristic in terms of the independent characteristics. The approach has been evaluated using a case study within the UWV.

Future work aims at making a more extensive taxonomy of analysis use cases. In this paper only a few examples were mentioned. Moreover, we would like to support the user in selecting the right use case using a questionnaire-based approach. This can be done by building on the current framework and implementation. Regarding improving the correlation accuracy, we also plan to investigate random decision forests, where several decision trees are built in multiple steps. We also acknowledge the limitations of our framework when the dependent characteristic is numerical. The results are not very "stable": a small change in how the characteristic is discretized may have large repercussions on the resulting decision tree. We also plan to investigate solutions to overcome this problem.

References

1. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: On the Move to Meaningful Internet Systems: OTM 2012. Volume 7565 of LNCS. Springer Berlin Heidelberg (2012) 287–304

2. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Information Systems **36**(2) (2011) 450–475
3. Lakshmanan, G., Shamsi, D., Doganata, Y., Unuvar, M., Khalaf, R.: A markov prediction model for data-driven semi-structured business processes. Knowledge and Information Systems (2013) 1–30
4. Ghattas, J., Soffer, P., Peleg, M.: Improving business process decision making based on past experience. Decision Support Systems **59** (2014) 93 – 107
5. Kim, A., Obregon, J., Jung, J.Y.: Constructing decision trees from process logs for performer recommendation. In: Proceedings of 2013 Business Process Management Workshops. Volume 171 of LNBIP., Springer (2014) 224–236
6. Conforti, R., de Leoni, M., La Rosa, M., van der Aalst, W.M.P.: Supporting risk-informed decisions during business process execution. In: Proceedings of the 25th International Conference on Advanced Information Systems Engineering (CAISE'13). Volume 7908 of LNCS., Springer-Verlag (2013) 116–132
7. Maggi, F.M., Francescomarino, C.D., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Proceedings of the 26th International Conference on Advanced Information Systems Engineering (CAiSE 2014). Volume 8484 of LNCS. (2014) 457–472
8. Dohmen, A., Moormann, J.: Identifying Drivers of Inefficiency in Business Processes: A DEA and Data Mining Perspective. In: Enterprise, Business-Process and Information Systems Modeling. Volume 50 of LNBIP. Springer Berlin Heidelberg (2010) 120–132
9. Zeng, L., Lingenfelder, C., Lei, H., Chang, H.: Event-driven quality of service prediction. In: Proceedings of the 8th International Conference of Service-Oriented Computing (ICSOC 2008). Volume 5364 of LNCS. Springer Berlin Heidelberg (2008) 147–161
10. van der Aalst, W.M.P., Dustdar, S.: Process mining put into context. IEEE Internet Computing **16**(1) (2012) 82–86
11. Sutrisnowati, R.A., Bae, H., Park, J., Ha, B.H.: Learning bayesian network from event logs using mutual information test. In: Proceedings of the 6th International Conference on Service-Oriented Computing and Applications (SOCA). (2013) 356–360
12. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communication **7**(1) (1994) 39–59
13. Mitchell, T.M.: Machine Learning. 1 edn. McGraw-Hill, Inc., New York, NY, USA (1997)
14. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Proceedings of the Twelfth International Conference on Machine Learning (ICML'95), Morgan Kaufmann (1995) 194–202
15. Nakatumba, J.: Resource-Aware Business Process Management: Analysis and Support. PhD thesis, Eindhoven University of Technology (2014) ISBN: 978-90-386-3472-2.
16. van der Aalst, W.M.P., Beer, H.T., van Dongen, B.F.: Process mining and verification of properties: An approach based on temporal logic. In: Conference On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE. Volume 3760 of LNCS., Springer Berlin Heidelberg (2005) 130–147
17. de Leoni, M., van der Aalst, W.M.P.: Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming. In: Proceedings of the 11th International Conference on Business Process Management (BPM'13). Volume 8094 of LNCS., Springer-Verlag (2013) 113–129
18. de Leoni, M., Maggi, F.M., van der Aalst, W.M.P.: An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data. Information Systems (2014) To appear. Doi: 10.1016/j.is.2013.12.005.
19. Rozinat, A., van der Aalst, W.M.P.: Decision Mining in Prom. In: Proceedings of the 4th International Conference on Business Process Management (BPM'06). LNCS, Springer-Verlag (2006) 420–425