Diagnostics in compliance checking

Gromov, V.V.

Award date:
2014

Technische Universiteit **Eindhoven** University of Technology

Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

Diagnostics in compliance checking

Master Thesis

Vladimir Gromov

Supervisors:
Prof.dr.ir. Wil M.P. van der Aalst
Dr. Martijn C. Schut
Elham Ramezani Taghiabadi MSc

Public version
(some sections were excluded due to confidentiality issues)

Eindhoven, January 2014

Abstract

Nowadays, information systems in organizations are recording nearly all actions performed by users in the form of event logs. Event logs can be analyzed using process mining techniques. The techniques allow to check compliance of an event log to a specified behavior. A specification may include control-flow, data, organization and time perspectives. Unfortunately, the output of such checking does not provide enough insight to understand if there is a meaningful relation between violations and the context. User would have to inspect the log manually, or using different means to discover it.

In this master thesis we developed an approach to investigate the context information of the violations discovered in event logs during the compliance checking. The approach takes two steps. First, a general overview of the problems in the log is obtained. Second, a specific problem is examined in detail in order to provide deeper insights. These steps were implemented as two plugins. Both plugins have been implemented using the process mining framework, ProM.

Our approach was evaluated through artificial and real-life data. The evaluation outcomes confirm that the approach allows us to identify the context related to the violations and the implemented software presents it in clear way.

Preface

This master thesis is the result of my graduation project which completes the master Computer Science & Engineering at the Eindhoven University of Technology. The project was performed under the supervision of members of the the Architecture of Information Systems group of the Mathematics and Computer Science department of Eindhoven University of Technology and conducted within the Forensics group of PricewaterhouseCoopers Advisory N.V.

First of all I would like to thank professor Wil van der Aalst and Martijn Schut for their guidance, advices and supports during my master project.

I would also like to thank Dr. Eric Verbeek and Dr. Massimiliano de Leoni for guidance with ProM Framework and Conformance Checking. Besides, I am grateful to Joos Buijs for creating a template for this thesis. Furthermore, I would specially thank Elham Ramezani Taghiabadi, for her patience during our discussions. Finally, my thanks go out to my parents for the huge support they gave me throughout my education.

Vladimir Gromov January, 2014

Contents

Chapter 1

Introduction

This master thesis is the result of a graduation project conducted as part of the Computer Science and Engineering master program at Eindhoven University of Technology (TU/e). The project was carried out within the Forensics group of PricewaterhouseCoopers (PwC) under supervision of members of the Architecture of Information Systems (AIS) group of the Mathematics and Computer Science department of TU/e.

In this thesis we investigate the problem of obtaining and visualizing diagnostics that give insights into violations in business process executions recorded in the form of event logs. To achieve this, we specify compliance requirements in terms of a process model that also describes data flow (a Petri Net with Data [18] to be precise). To check if business processes are executed in accordance with the specified requirements, we apply conformance checking techniques. We use the result of compliance analysis to provide diagnostic information about the discovered violations to business users. A violation can be about different perspectives of a business process, including control-flow, time, data, and resource.

In this chapter we introduce the problem in more detail. In Section 1.1 we give the context of this thesis. Section 1.2 describes the addressed problem. Section 1.3 defines the scope of the thesis. Finally, Section 1.4 outlines structure of the thesis.

This is the public version of the thesis: some sections were removed due to confidentiality issues.

1.1 Thesis Context

Contemporary businesses are becoming more and more dependent on information technology. Companies are increasingly introducing information systems in order to support their business processes. Such systems help in managing all aspects of companies' performance, including finance, procurement, sales, production, human resources or customer relationship management. Despite this increase in automation, most processes are still to a great extent manual and error prone. Organizations must comply with a continuously increasing set of regulations and laws. Therefore, it is becoming harder to monitor errors, while their cost might be considerable, due to either loss of income, or penalties. ISO 9001 is an example of regulation. Non-compliance with this standard does not lead to penalties, but some customers might require it from their suppliers. Generally speaking, non-compliance to some specific rules might indicate fraud, which in turn can have considerable negative financial impact. According to PwC Crime Survey 2011 [3] involving 250 Dutch companies, antitrust (cartel) agreements or financial fraud caused a direct loss of more than €500,000 in a quarter of cases.

Enterprise information systems, like SAP Business Suite, Oracle E-Business Suite and Microsoft Dynamics NAV, are recording all kinds of actions performed by users. Regardless of the system used, the recorded information can usually be represented in form of a unified event log. The fact that event logs are usually readily available allows automated compliance checking to be performed and provides the possibility for further analysis to give more insight and diagnostics if something is wrong.

An event log may include for each event the following data: event name, originator (responsible person or system), time of occurrence and optionally context information, such as purchased amount (in a procurement process), produced goods for production, client ID for sales, etc. This information provides the possibility to analyze a business process from different angles.

Relations between events (like ordering and (non-)coexistence) reflect the Control-Flow perspective of the process; time of their occurrence and the delay between them represent the time perspective; information about the originators determines organizational perspective, while all other context information shows the data perspective of a process [5]. Such event logs can serve as the starting point for analyzing compliance of a recorded process to specific regulations.

Compliance requirements may restrict different perspectives of business processes. Hence, it is necessary to check compliance of a business process from different perspectives and provide comprehensive diagnostics. Furthermore, to analyze a violation in deep, we need to consider the context where the violation occurred.

Assume we are observing a procurement process of some company. According to its regulations, a payment should be sent to a supplier only after receipt of purchased goods. If, after checking this rule we identify a number of cases where payment is performed before goods have been received, their context might allow us to understand the violation better. If such payment is always performed by the same person, it is likely to indicate fraud or the incompetence of this person. If such violations occur for the same supplier, it might be due to peculiarity of the supplier. It is possible that there are no dependencies between the violations and their context. In this case, the reason may be lack of communication or enforcement of the company's regulations. Thus, comparing the context of a violating event to that of a non-violating one allows us to understand the root cause of the problem or find out the direction for further investigation.

Auditors and compliance experts are usually less familiar with formal process mining methods and techniques, so the diagnostics must be presented to them in an intuitive way. Also, since software is unable to justify whether a violation indicates fraud or a simple error, it should provide means for deeper analysis of critical in order to allow experts to obtain a complete picture. Unfortunately, software is not capable of determining the cause of a violation. However, detailed diagnostics would help analysts to perform an investigation and discover it.

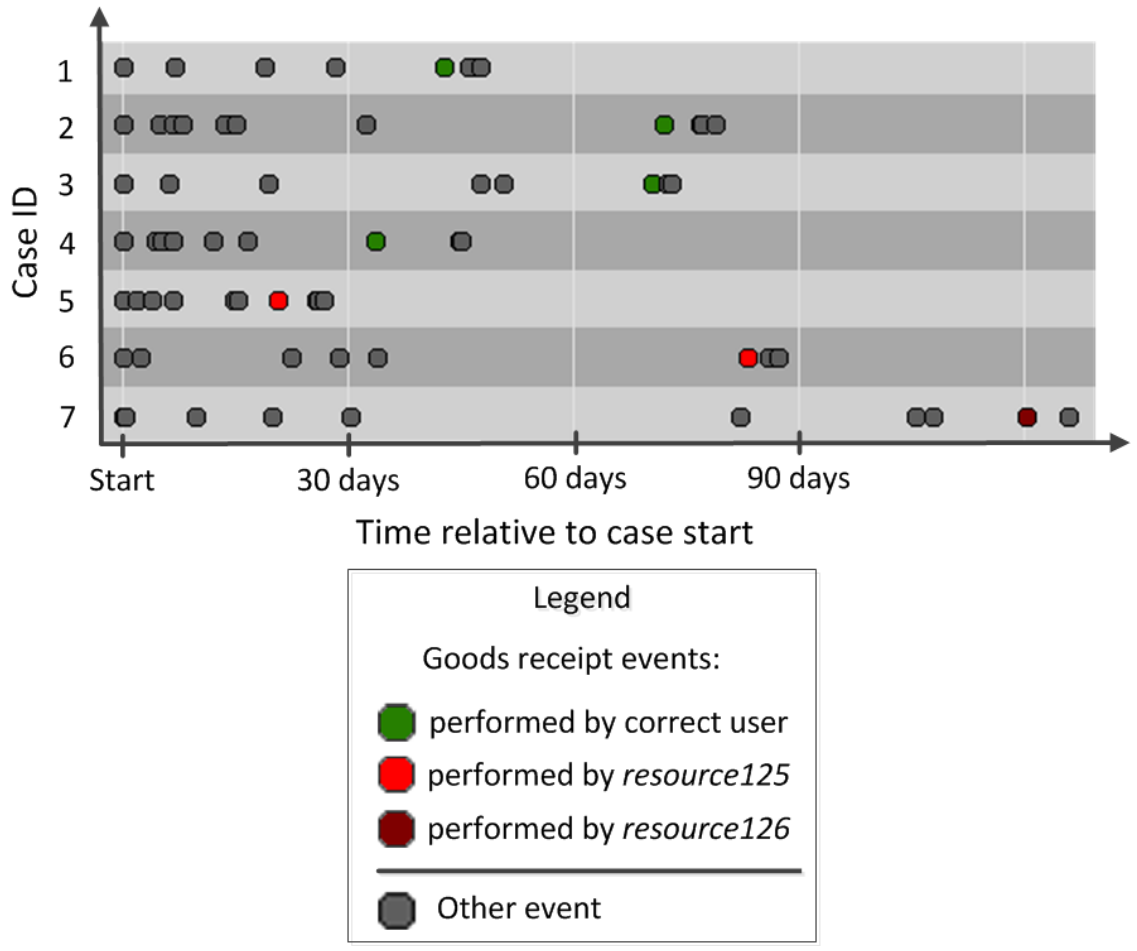


Figure 1.1: Example log excerpt

1.2 Project Objective

In Section 1.1, we have introduced the thesis context elaborating on the challenges that modern businesses are facing. In this section, we define the objective of this thesis.

We discussed that ensuring compliance is becoming more and more important for modern businesses. In the area of process mining, different techniques [5] have been developed for compliance checking of business processes. These techniques check the compliance of a business process recorded in the form of an event log with different types of compliance rules. Existing compliance checking techniques are capable of verifying all perspectives of a process, but the output does not integrate results from checking different perspectives, forcing the user to observe them separately. Besides, they are not aimed at identifying dependencies between discovered problems and the context. This makes it much harder for the user to understand why violations occur and where a deeper investigation should be directed.

As was mentioned, people responsible for examining business processes usually do not have a strong technical background. This raises a critical need for a comprehensible representation of compliance diagnostics, which is generally not provided.

The described problems motivate the following research objective:

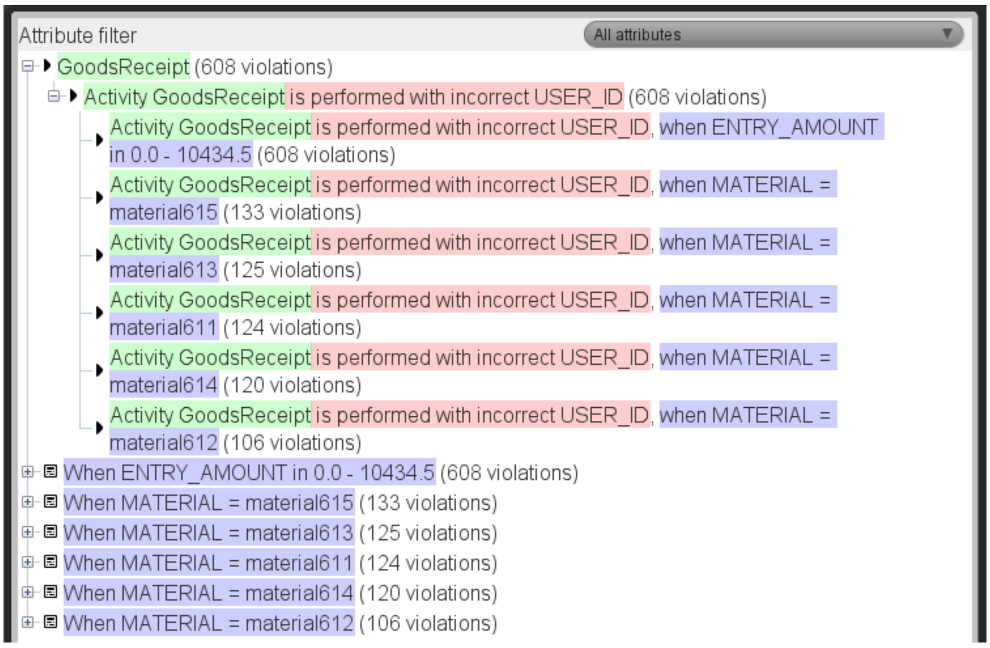
Develop a systematic approach to get insights about the violations detected during compliance checking through the analysis of context-related information available in the business process and present the obtained results in a way understandable for users not having deep technical knowledge.

Our research problem can be divided into four sub-problems:

1. Given an event log of a process and a set of compliance rules, obtain data about present violations using previously developed compliance checking techniques.
2. Given information about violations in the process, provide a list of discovered problems and related diagnostics as well as statistics aggregated for each activity or context element.
3. Given a specific problem, discover a way to obtain deep insights into the connected context and provide detailed diagnostics.
4. Develop a way to visualize the resulting diagnostics in a way understandable for users without a technical background.

In order to illustrate our research objectives, we describe an example. In Figure 1.1, an excerpt of a dotted chart is shown, which demonstrates an event log of a procurement process. According to a compliance rule, 'Goods receipt' can be performed only by users with ID: *resource71, resource72, . . . ,resource100.*

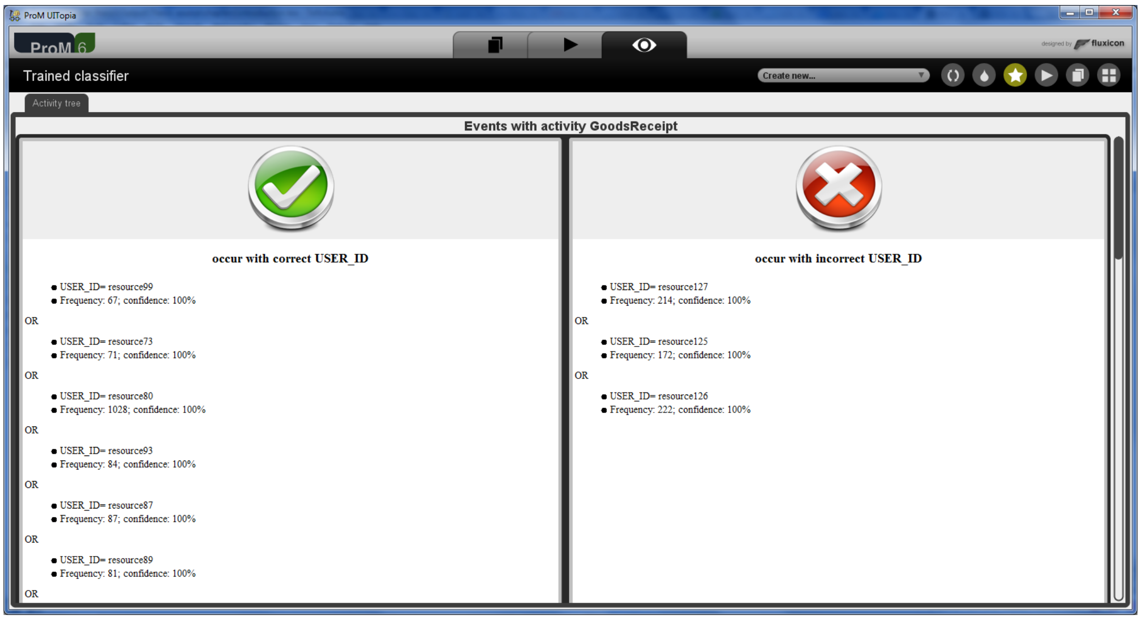
After applying our approach and tools to this event log, we obtained the compliance report, shown in Figure 1.2. The report shows that there are multiple occurrences of the a problem being 'Goods receipt is performed by a wrong user'. Besides, it briefly lists other context elements available in the event log that were related to the problem.


Attribute filter
All attributes

- GoodsReceipt (608 violations)
  - Activity GoodsReceipt is performed with incorrect USER_ID (608 violations)
    - Activity GoodsReceipt is performed with incorrect USER_ID, when ENTRY_AMOUNT in 0.0 - 10434.5 (608 violations)
    - Activity GoodsReceipt is performed with incorrect USER_ID, when MATERIAL = material615 (133 violations)
    - Activity GoodsReceipt is performed with incorrect USER_ID, when MATERIAL = material613 (125 violations)
    - Activity GoodsReceipt is performed with incorrect USER_ID, when MATERIAL = material611 (124 violations)
    - Activity GoodsReceipt is performed with incorrect USER_ID, when MATERIAL = material614 (120 violations)
    - Activity GoodsReceipt is performed with incorrect USER_ID, when MATERIAL = material612 (106 violations)
- When ENTRY_AMOUNT in 0.0 - 10434.5 (608 violations)
- When MATERIAL = material615 (133 violations)
- When MATERIAL = material613 (125 violations)
- When MATERIAL = material611 (124 violations)
- When MATERIAL = material614 (120 violations)
- When MATERIAL = material612 (106 violations)

Figure 1.2: List of discovered problems.

If we further analyze the identified problem in order to discover the root cause of it, we would obtain the report shown in Figure 1.3. It shows that the problem is most likely to occur when specific users are in charge (*resource125, resource126,resource127* to be precise).



Events with activity GoodsReceipt

occur with correct USER_ID

- USER_ID= resource99
- Frequency: 67; confidence: 100%

OR

- USER_ID= resource73
- Frequency: 71; confidence: 100%

OR

- USER_ID= resource80
- Frequency: 1028; confidence: 100%

OR

- USER_ID= resource93
- Frequency: 84; confidence: 100%

OR

- USER_ID= resource87
- Frequency: 87; confidence: 100%

OR

- USER_ID= resource89
- Frequency: 81; confidence: 100%

OR

occur with incorrect USER_ID

- USER_ID= resource127
- Frequency: 214; confidence: 100%

OR

- USER_ID= resource125
- Frequency: 172; confidence: 100%

OR

- USER_ID= resource126
- Frequency: 222; confidence: 100%

Figure 1.3: Root cause of the violation. Users *resource125, resource126, resource127* are responsible for the violating events

1.3 Research Scope and Methodology

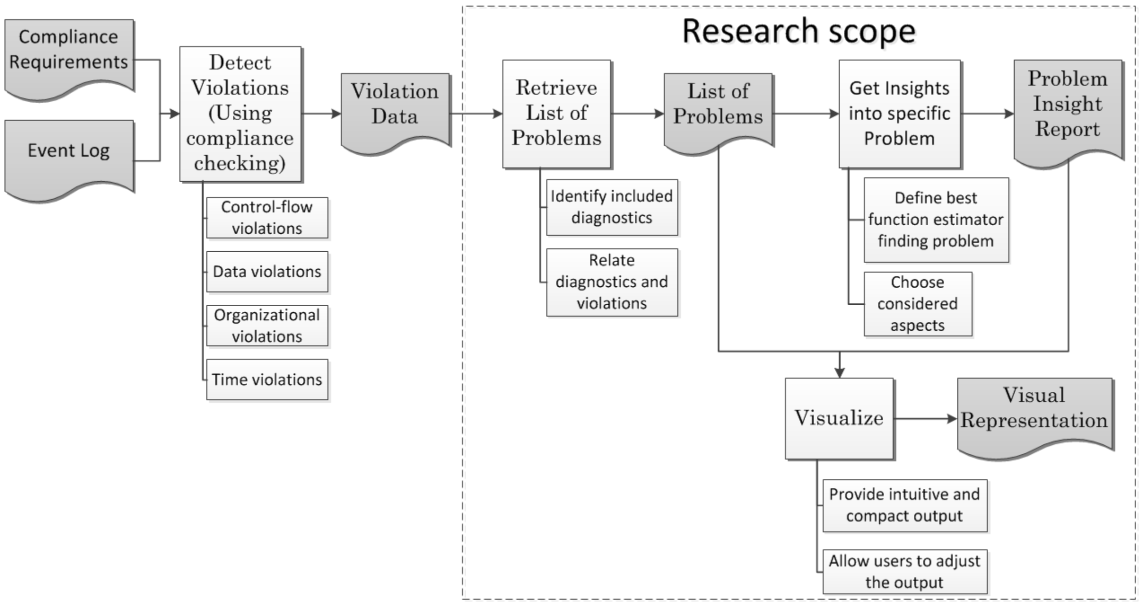

Research scope
Compliance Requirements
Event Log
Detect Violations (Using compliance checking)
Control-flow violations
Data violations
Organizational violations
Time violations
Violation Data
Retrieve List of Problems
Identify included diagnostics
Relate diagnostics and violations
List of Problems
Get Insights into specific Problem
Define best function estimator finding problem
Choose considered aspects
Problem Insight Report
Visualize
Provide intuitive and compact output
Allow users to adjust the output
Visual Representation

Figure 1.4: Research methodology

We have defined the research problem and divided it into four sub-problems. In this section we define the research scope of the thesis and the steps we have to carry out (Figure 1.4).

It is assumed that *event logs* are available as an input. Besides, *compliance requirements* should also be provided. These are considered as input for *compliance checking* techniques we deploy, used for obtaining *violation data* present in the log. We need to choose best compliance checking techniques available in order to discover violations. Since compliance requirements may restrict different perspectives of a business process, we need to identify violations in all process perspectives including: control-flow, time, resource, and data perspectives.

The *violation data* serve as a starting point for obtaining the *list of problems* and aggregating violations. Such a list reports on the discovered problems along with brief diagnostic information regarding the context of the problems. In order to provide the list of all violations with diagnostic information about them, we need to explore the information available in the log. Moreover, it is necessary to investigate how well the diagnostics explain the violations by quantifying the relation between violations and identified diagnostics.

Using the description of a single problem as input, a deeper analysis is performed in the *obtain insight* step, in order to provide detailed diagnostics regarding the specified problem. This information helps in understanding the root cause of the problem. To fulfill this, occurrences of a violating event where the problem is present should be compared to compliant events where the problem is absent. The comparison is done by finding the best estimator of a function, where observed input consists of attributes of the compared event and observed output is an indication whether the event experienced the analyzed problem.

Finally, the obtained information should be *visualized* in an appropriate way for users without technical knowledge. Obviously, the visualization of this information should be intuitive and compact. Moreover, in order to allow users to gain most from the approach, the visualization should provide the option to adjust the output. In this way, users would be able to emphasize the information most important in their opinion.

The scope of this research is bounded by retrieving the list of violations, obtaining insights into a single problem, and visualizing the results.

1.4 Thesis Structure

The thesis is structured as follows:

Basic terms and concepts used throughout the thesis are introduced and literature study on related work is described (Chapter 2).
Concepts from process mining with focus on compliance checking are introduced. Also, some concepts required for quantifying relations between violations and underlying context are described. Finally, an overview of related work on each part of the thesis is given.

Describing compliance checking techniques required for obtaining alignments to identify violations (Chapter 3).
Since compliance requirements may restrict different perspectives of a business process, in this chapter we discuss the compliance checking techniques we deployed for analysis of all process perspectives including: control-flow, time, resource, and data. In this chapter we show how we integrate all compliance results from different perspectives.

Explaining ways of identifying and ranking problems discovered during compliance checking (Chapter 4).
After identifying all violations, we need to summarize all discovered problems and define a way to rank them.

Investigating specific problems (Chapter 5).
Once a list of problems is obtained, we investigate the root cause of a single problem from the list.

Implementation in ProM (Chapter 6).
The architecture of our analysis software and the design decisions made are described.

Evaluating the implementation and demonstrating experimental results (Chapter 7).
Correctness of the approach is verified using an artificial event log. Furthermore understandability and applicability of the tools are evaluated by users. We applied our technique to a real-life event log. We will discuss the results of our analysis in this chapter.

Conclusion (Chapter 8).
The entire approach is summarized in this chapter. We will discuss whether project goals were achieved. Also, limitations of the developed approach along with future work are explained.

Chapter 2

Preliminaries

We have defined our research goal and scope in the previous chapter. Before describing our approach in more detail, we would like to discuss the concepts that our work is built on.

In Section 2.1 we are describing process mining notions used in the approach. First, a general overview of process mining is given, after which we describe *event logs*, *specified behavior* and *alignments* in detail. In Section 2.2 we introduce the notion of *conditional-probability increment ratio*, using the notions of *itemsets* and *transactions*. Section 2.3 gives an overview of the related work.

2.1 Process Mining

The idea of process mining is to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in today's systems [5].

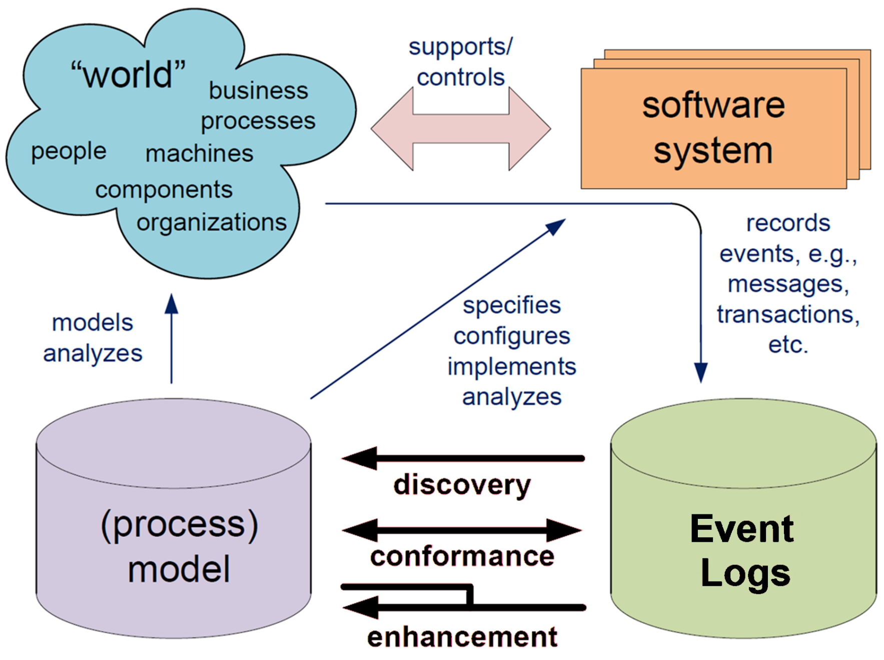


Figure 2.1: Positioning of the three main types of process mining: discovery, conformance, and enhancement [5].

In Figure 2.1, the three main types of process mining are positioned. According to the figure, information about processes is collected by software systems into event logs, which are in turn analyzed with process mining techniques.

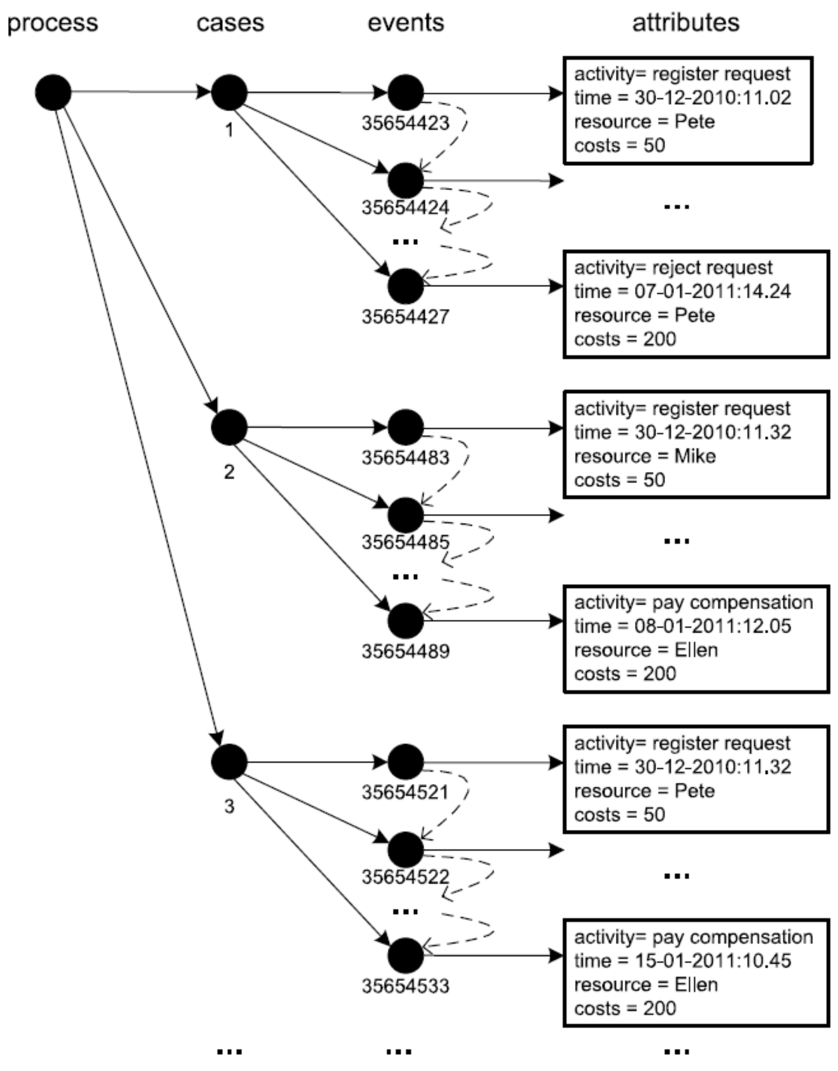


Figure 2.2: Structure of event logs [5].

According to [5], there are three types of process mining. The first type is *discovery.* A discovery technique takes an event log and produces some kind of process model that describes the observed behavior without using any a-priori information. If the event log contains information about resources, one can also discover resource-related models, e.g., a social network showing how people work together in an organization [8].

The second type of process mining is *conformance checking.* Here, an existing process model is compared with an event log of the same process. For instance, there may be a process model indicating that purchase orders of more than one million euros require two checks. Analysis of an event log would show whether this rule has been followed or not. Another example is checking the so-called "four-eyes" principle: particular activities should not be executed by one and the same person. By scanning an event log using a model specifying these requirements, one can discover potential cases of fraud. Hence, conformance checking may be used to detect, locate and explain violations, and to measure their severity. In this thesis the focus is on conformance checking, i.e. the second type of process mining in Figure 2.1.

The third type of process mining is *enhancement*. Here, the idea is to extend or improve an existing process model using information about the actual process recorded in some event log.

Any of the process mining techniques takes an event log as an input, which is expected to have a certain structure. The structure is depicted in Figure 2.2. Generally, event data from an event log corresponds to a process. A process is composed of cases or completed process instances. In turn, a case consists of sequence of events, called a trace. An event has a name and various attributes. A formal definition of event log is given in Section 2.1.1.

Event data is not usually recorded for process mining purposes and thus may have varying formats. Therefore, a standard format is required to perform analysis. One such format is XES[1]. This format is selected as the standard format for event logs by the IEEE Task Force on Process Mining. Besides, it has support from popular process mining tools such as ProM[2] and Disco[3].

2.1.1 Event Log

Let E be the finite set of all event names $E = \{ename_1, \dots, ename_n\}$. Let $E^{\gg}$ be a set of event names including the explicit *no event*, denoted as $\gg$, i.e. $E^{\gg} = E \cup \{\gg\}$.
Let V_L be a set of all *attributes* of an event (including timestamp and resource), excluding the event name.
We distinguish a set $V_L^{time} \subseteq V_L$ of attributes bearing time information and a $V_L^{context} \subseteq V_L$ of all other attributes, such that $V_L^{context} = V_L \setminus V_L^{time}$.
Let G_v be a set of all possible *values* for an attribute $v \in V_L$
Let U_L be a superset of all G_v, i.e. $U_L = \bigcup\limits_{v \in V} G_v$.
Each *event* e is a tuple $(ename, ((v_1, u_1), \dots, (v_m, u_m)))$, where:

- (v_i, u_i) is a pair describing an attribute and its value, called *attribute assignment*, where $v_i \in V_L$ is an attribute, $u_i \in G_{v_i}$, is a value of attribute v_i
- *ename* is an event name, $ename \in E$.

σ_L is a *trace*, a sequence of *events* $\langle e_1, \dots, e_n \rangle$.
Let L be an *event log*, a multiset of traces, i.e. $L \in \mathcal{B}((E \times (V_L \times U_L)^m)^*)$[4]. All events in the log are supposed to have the same number of attributes.
Let X be a multiset of all events in L, i.e. $X \in \mathcal{B}(E \times (V_L \times U_L)^m)$. Let $X^{\gg}$ be a multiset of all events in L including an explicit *no event*, i.e. $X^{\gg} \in \mathcal{B}(E^{\gg} \times (V_L \times U_L)^m)$. This notion will used further in Section 2.1.3.
Let $val_L : X^{\gg} \times V_L \rightarrow G_v$ be a function that maps an event and attribute to the attribute's value. Given an event x with attribute v and its value u, $val_L(x, v) = u$

Example 1. *In order to illustrate the above mentioned notions, let us consider a simplified example. An insurance company has activated a process to deal with compensations requested by clients. The process starts with a request initiated by any of the employees. After that the documents are checked in any order and the request is examined. Requests for an amount above or equal to €5000 should be examined thoroughly, while those for lower amounts should be examined casually. The casual examination can be done either by Mike or by Ellen, while a thorough examination is done by Sara. The documents can be checked either by Pete or by Sean. After that the decision is made by Sara and the request can be either re-initiated, payed or declined.* An example of a log trace for such process could be:

$\langle e_1 = (initiate\ request, ((AR, 3000), (Ex, Pete), (TS, 0))),$
$e_2 = (examine\ casually, ((AR, 3000), (Ex, Ellen), (TS, 1000))),$
$e_3 = (check\ documents, ((AR, 3000), (Ex, Pete), (TS, 2000))),$
$e_4 = (decide, ((AR, 3000), (Ex, Sara), (TS, 3000))),$

[1] A detailed description of the XES format can be found on `http://www.xes-standard.org/`
[2] `http://www.promtools.org/prom6/`
[3] `http://fluxicon.com/disco/`
[4] $\mathcal{B}(A)$ is the set of all multisets over A

$e_5 = (\textit{pay compensation}, ((AR, 3000), (Ex, Pete), (TS, 4000)))\rangle$

The set of event names $E = \{\textit{initiate request}, \textit{examine thoroughly}, \textit{examine casually}, \textit{check documents}, \textit{decide}, \textit{pay compensation}\}$. Each event has 3 attributes: AR, which denotes the size of the requested compensation, Ex, which stands for Executor and denotes the activity executor, and TS which shows time of the event relative to the first event of the case in seconds, i.e. $V_L = \{AR, Ex, TS\}$, $V_L^{context} = \{AR, Ex\}$, $V_L^{time} = \{TS\}$. The possible values for AR are $G_{AR} = \mathbb{Q}^+$, where $\mathbb{Q}^+ = \{q \in \mathbb{Q} \mid q > 0\}$. $G_{Ex} = \{Pete, Ellen, Sara, Sean, Mike\}$. $G_{TS} = \mathbb{N}_0$ ($\mathbb{N}_0$ is the set of natural numbers including 0). Thus, $U_L = \mathbb{Q}^+ \cup \{Pete, Ellen, Sara, Sean, Mike\} \cup \mathbb{N}_0$. The values of the function val_L for the event e_1 are: $val_L(e_1, AR) = 3000$; $val_L(e_1, Ex) = Pete$.

2.1.2 Specified Behavior

Let V_M be the set of *attributes* defined in the specification.
Let A be the finite set of all *activity names* $\{aname_1, \ldots, aname_n\}$. Let $A^{\gg}$ be a set of activity names including explicit *no activity*, denoted as $\gg$, i.e. $A^{\gg} = A \cup \{\gg\}$
Similarly to an event, an *activity* a is a tuple $a = (name, ((v_1, H_1), \ldots, (v_l, H_l)))$, where:

- (v_i, H_i) is a pair of context attribute and a set of admissible values for this activity, including time and resource attributes, where
- $v_i \in V_M$ is an attribute,
- H_i is the set of allowed values for attribute v_i,
- *aname* is an activity name, $aname \in A$.

Let U_M be a set of all H_i, i.e. $U_M = \{H_1, ..., H_l\}$, where l is the total number of data attributes.
Let σ_S be a *sequence of activities* $\langle a_1, \ldots, a_k\rangle$ which is compliant with a certain compliance rule. We define S as the set of all sequences which are compliant with a certain compliance rule, $S \subseteq (A \times (V_M \times U_M)^l)^*$.
Let Y be a set of all activities in S, i.e. $Y \subseteq (A \times (V_M \times U_M)^l)$ and $Y \neq \varnothing$. Let $Y^{\gg}$ be a set of all activities, including explicit *no activity*, $Y^{\gg} \subseteq (A^{\gg} \times (V_M \times U_M)^l)$ and $Y^{\gg} \neq \varnothing$.
Let Y_{inv} be a set of invisible activities, $Y_{inv} \subseteq Y$. The difference between the invisible activity and visible will be explained further in Section 2.1.3
Let $dom(a, v)$ be a function that assigns the values admissible for attribute $v \in V_M$ allowed by activity a, $dom(a, v) = H_v^a$. H_v^a is the domain of v for activity a.

The specification S is normally very large varying over all admissible sequences of activities in combination with admissible attribute values. The set S is the semantic collection of all compliant traces. We use a Petri Net with Data N to specify the admissible behavior S in a compact form, such that S is the set of all terminating runs of N.

Definition (DP-net). A *Petri Net with Data* (DP-net) $N = (P, T, F, V_M, R, W, G)$ consists of:

- P is a set of *places*;
- T is a set of *transitions*;
- T_{inv} is a set of *invisible transitions*, $T_{inv} \subseteq T$;
- $F \subseteq (P \times T) \cup (T \times P)$ is the *follow relation* describing the arcs between places and transitions (and between transitions and places);
- a set V_M of attributes;
- a guard function $G : T \to \mathcal{G}_{V_M}$ that associates a guard to a transition. Guards restrict log attributes. Based on them, admissible values for combination of activities and attributes are defined.

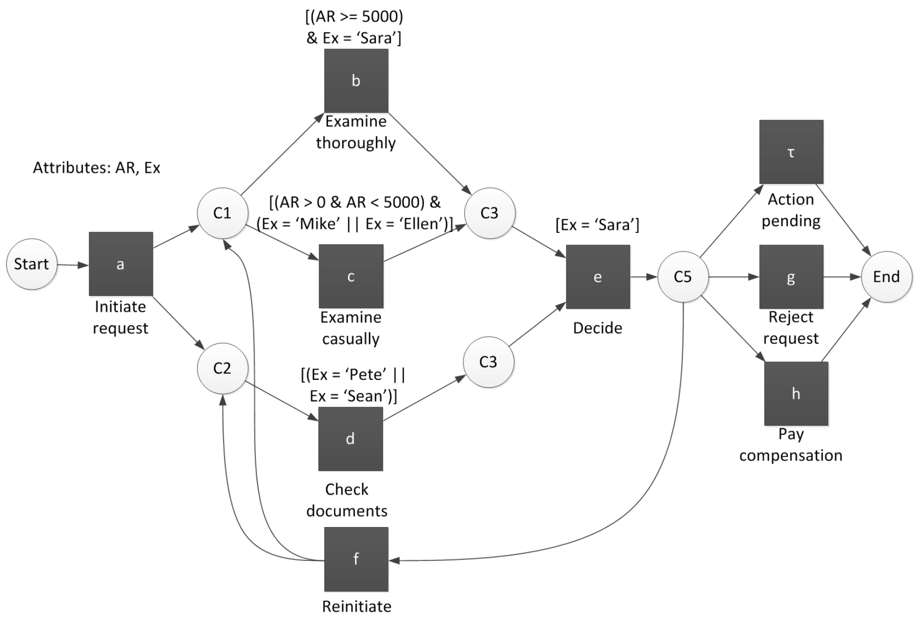


Figure 2.3: Petri Net with Data N_S example

An example of a Petri Net with Data N_S describing a simplified compensation application process is shown in Figure 2.3.

In the example:

- the set of places $P = \{start, c1, c2, c3, c4, c5, end\}$;
- the set of transitions $T = \{a, b, c, d, e, f, g, h, f, \tau\}$;
- the set of invisible transitions $T_{inv} = \{\tau\}$;
- the set of arcs $F = ((start, a), (a, c1), (a, c2), (c1, b), (c1, c), (c2, d), (b, c3), (c, c3),$ $(d, c4), (c3, e), (c4, e), (e, c5), (c5, g), (c5, h), (c5, f), (g, end), (h, end), (f, c1), (f, c2));$
- the set of attributes $V_M = \{AR, Ex\}$;
- function $G(b) = [(AR >= 5000)\,\&\,(Ex = Sara)]$, $G(c) = [(AR > 0\,\&\,AR < 5000)\,\&\,(Ex = Mike\ ||\ Ex = Ellen)]$, $G(e) = [Ex = Sara]$, $G(d) = [(Ex = Pete\ ||\ Ex = Sean)]$

One of the possible traces is $\sigma_S = \langle(a, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))),$
$(c, ((AR, \{q \in \mathbb{Q}^+ \mid q < 5000\}), (Ex, \{Mike, Ellen\}))), (d, ((AR, \mathbb{Q}^+), (Ex, \{Pete, Sean\})))$
$(e, ((AR, \mathbb{Q}^+), (Ex, \{Sara\}))), (g, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U})))\rangle$, where $\mathbb{U}$ is a universal set.

If a DP-net does not define a guard in some transition for an attribute v_i, then its *allowed values set* is considered to be *any* for the corresponding activity.

2.1.3 Alignments

Let $em : E \nrightarrow A$ be a mapping function that maps an event name $ename \in E$ to an activity name $aname \in A$ so that $em(ename) = aname$.
An *alignment* γ is a sequence of Data-Aware Steps, $\gamma = \langle z_1, \ldots, z_n\rangle$ such that the projection on the first element (ignoring events with $ename = \gg$) yields $\sigma_L \in L$ and the projection on the second element yields $\sigma_S \in S$ (ignoring activities with $aname = \gg$).
A *Data-Aware Replay Result DR* that relates events in the log to activities in the specification, is a multiset of alignments. *DR* is a multi-set because an event log may contain the same log trace σ_L multiple times, potentially resulting in multiple identical alignments.

An *alignment* is built by relating all traces σ_L in event log L to a specification trace σ_S by pairing events in σ_L to activities in $\sigma_S \in S$.
A *data aware replay step* z is a pair $(x, y), x \in X^{\gg}, y \in Y^{\gg}, z \in X^{\gg} \times Y^{\gg} \setminus \{(x, y) \mid x.ename = \gg \wedge y.aname = \gg\}$. An alignment is built such that for each $z = (x, y)$ $em(x.ename) = y.aname$ if:
$x.ename \neq \gg$ and $y.aname \neq \gg$
Or
$x.ename = \gg$ if $y \in Y_{inv}$,
i.e. invisible activities do not have corresponding events in the log. If $z = (x, y)$ and $x.ename = \gg$, x attribute values are the same as for the previous event.
Let Z be a *multiset of all Data-Aware Steps* in the Data-Aware Replay Result DR of log L to specification S. $Z \in \mathcal{B}((E^{\gg} \times (V_L \times U_L)^m) \times (A^{\gg} \times (V_M \times U_M)^l))$.

Let us consider an example trace of a log of the above mentioned compensation handling process. Hereafter we do not include the *TS* attribute in the examples, since it is insignificant for explanation of the concepts. The event names are encoded with the corresponding transition names:
$\sigma_{L1} = \langle (a, ((AR, 3000), (Ex, Pete))), (c, ((AR, 3000), (Ex, Mike))),$
$(d, ((AR, 3000), (Ex, Pete))), (e, ((AR, 3000), (Ex, Sara))) \rangle$,
and a specification is described by the Petri Net N_S in Figure 2.3. This trace corresponds to the specified trace σ_{S1} perfectly. The alignment γ_1 between them contains the step with *ename* = *no event* ($\gg$), but this is not considered to be a violation, since activity τ is invisible, i.e. $\tau \in Y_{inv}$. The presence of this activity in the alignment indicates that the case is not yet finished, i.e. action has not yet been taken and the request is pending. The obtained alignment is shown in Table 2.1. Please note that events with $ename = \gg$ derive attribute values from the one previous to it or, if such event is the first in a trace, from the nearest event with $ename \neq \gg$.

Table 2.1: Alignment γ_1

γ_1	
σ_{L1}	σ_{S1}
$a, ((AR, 3000), (Ex, Pete))$	$a, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
$c, ((AR, 3000), (Ex, Mike))$	$c, ((AR, \{q \in \mathbb{Q}^+ \mid q < 5000\}), (Ex, \{Mike, Ellen\}))$
$d, ((AR, 3000), (Ex, Pete))$	$d, ((AR, \mathbb{Q}^+), (Ex, \{Pete, Sean\}))$
$e, ((AR, 3000), (Ex, Sara))$	$e, ((AR, \mathbb{Q}^+), (Ex, \{Sara\}))$
$\gg, ((AR, 3000), (Ex, Sara))$	$\tau, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$

However, an observed trace in a log may deviate from specified behavior. For example, let us consider another log trace of the same process:
$\sigma_{L2} = \langle (a, ((AR, 6000), (Ex, Pete))), (c, ((AR, 6000), (Ex, Pete))),$
$(b, ((AR, 6000), (Ex, Ellen))), (d, ((AR, 6000), (Ex, Mike))), (g, ((AR, 6000), (Ex, Pete))) \rangle$
This trace is not specified by N_S. The closest trace to σ_{L2} is σ_{S2} chosen in the alignment γ_2, shown in Table 2.2.
The specified trace σ_{S2} is chosen in order to minimize the number of violations.
In comparing a trace in an event log with a specification, the possible violations are:

- A *move on log* is a Data-Aware Step $z = (x, y)$, where $x.ename \neq \gg$ and $y.aname = \gg$;
- A *move on model*, which is considered a violation only for visible activities, is a Data-Aware Step $z = (x, y)$, where $x.ename = \gg$ and $y.aname \neq \gg$ and $y \notin Y_{inv}$;
- A *move with incorrect w* is a Data-Aware Step $z = (x, y)$, where $y.aname \neq \gg$, $x.v = y.w$, and $val_L(x.v) \notin dom(y.w)$.

Table 2.2: Alignment γ_2

γ_2	
σ_{L2}	σ_{S2}
$a, ((AR, 6000), (Ex, Pete))$	$a, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
$c, ((AR, 6000), (Ex, Pete))$	$\gg, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
$b, ((AR, 6000), (Ex, Ellen))$	$b, ((AR, \{q \in \mathbb{Q}^+ \mid q >= 5000\}), (Ex, \{Sara\}))$
$d, ((AR, 6000), (Ex, Mike))$	$d, ((AR, \mathbb{Q}^+), (Ex, \{Pete, Sean\}))$
$\gg, ((AR, 6000), (Ex, Mike))$	$e, ((AR, \mathbb{Q}^+), (Ex, \{Sara\}))$
$g, ((AR, 6000), (Ex, Pete))$	$g, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$

Apart from the violations, we distinguish *synchronous moves*. A *synchronous move* is a Data-Aware Step, that is neither a move on log, nor a move on model. It might still appear to be a move with incorrect w.

In our example γ_2 has 3 violations:

- Move on log for activity c;
- Move on model for activity e;
- Move with incorrect Ex for activities b and d;

2.1.4 Violations

We define several functions that indicate the presence of each type of violation, but first we have to define functions relating a Data-Aware Step to its components. The functions assume a given Data-Aware Replay Result DR, all Data-Aware Steps of which are contained in Z.
Given $z = (x, y) \in Z$:
$sname_L : Z \nrightarrow A$ - is a function which relates a move on log step to activity name, i.e. for $z = (x, y)$ and $y.aname = \gg$, $sname_L(z) = x.ename$
$sname_M : Z \nrightarrow A$ is a function, that relates non-move on log step in a replay result to an activity name, i.e. for $z = (x, y)$ and $y.aname \neq \gg$ $sname_M(z) = y.aname$
$sname(z) : Z \to A$

$$sname(z) = \begin{cases} sname_L(z), & \text{if } y.aname = \gg \\ sname_M(z), & otherwise \end{cases}$$

Given $z = (x, y)$, log attribute $v \in V_L$ and its value $u \in G_v$ so that $val_L(x, v) = u$ we define function:
$val : Z \times V_L \to U_L$ is a function that maps a step in a replay result to the value of the given attribute v in the log,

$$val(z, v) = u$$

Given $z = (x, y) \in Z$, specification attribute $w \in V_M$ with *allowed values* set H^y_w, so that $dom(y, w) = H^y_w$, in the specification we define the function:
$val_M : Z \times V_M \to U_M$ is a function that maps a step in a replay result to its allowed values according to the specification.

$$val_M(z, w) = H^y_w$$

We identify the following functions indicating the presence of problems given a Data-Aware Step $z = (x, y)$:
$P_{log.only} : Z \rightarrow \{true, false\}$ function, that indicates if there was a move on log

$$P_{log.only}(z) = \begin{cases} true, & \text{if } y.aname = \gg \\ false, & otherwise \end{cases}$$

$P_{model.only} : Z \rightarrow \{true, false\}$ function, that indicates if there was a move on model

$$P_{model.only}(z) = \begin{cases} true, & \text{if } x.ename = \gg \\ false, & otherwise \end{cases}$$

$P_{incorrect.attr} : Z \times V_M \rightarrow \{true, false\}$ function, that indicates if there was a move with incorrect attribute w.

$$P_{incorrect.attr}(z, w) = \begin{cases} true, & \text{if } \exists\, v[v \in V_L : v = w \wedge val(z, v) \notin val_M(z, w)] \\ false, & otherwise \end{cases}$$

2.2 Support and Conditional-Probability Increment Ratio

In this section we introduce itemsets and use them to define *CPIR* metric. We will modify this metric and use the modified version later on to relate and quantify the relation between violations and context data (in Section 4.3.3). Let $I = \{i_1, i_2, \ldots, i_N\}$ be a set of N distinct literals called items. I is called an *itemset*. Each item can be an activity name, or a pair of an event log attribute and its value, or a specification attribute, which indicates its violation, or an indication of move on log or move on model, i.e. $I = A \cup (V_L \times U_L) \cup \{log.only, model.only\} \cup V_M$.

Considering the example from Section 2.1.1 of the compensation handling process, our itemset could be $I = \{a, b, c, d, e, f, g, (Ex, Sara), (Ex, Sean), (Ex, Pete), (Ex, Ellen), (Ex, Mike), (AR, 5000), (AR, 6000), (AR, 3000), log.only, model.only, Ex, AR\}$ i.e. $i_1 = a, \ldots, i_8 = (Ex, Sara)$, etc. Then for each event the corresponding subset of our itemset can be found, e.g. for an event $e_1 = ((a, ((Ex, Sara), (AR, 5000)))$ the corresponding subset of the itemset would contain 3 items: event name a, $(Ex, Sara)$ and $(AR, 5000)$.

Given a Data-Aware Step $z = (x, y)$, where $x = (ename, ((v_1, u_1), \ldots, (v_l, u_l)))$, it is possible to find the corresponding subset T_z of the itemset I, $T_z \subseteq I$. The items of such set are:

$$\begin{cases} \{y.aname, (x.v_1, x.u_1), \ldots, (x.v_l, x.u_l), w_1, w_2, \ldots, w_m\}, & \text{if } x.ename \neq\gg \& y.aname \neq \gg \\ \{em(x.ename), (x.v_1, x.u_1), \ldots, (x.v_l, x.u_l), log.only\}, & \text{if } x.ename \neq\gg \& y.aname = \gg \\ \{y.aname, (x.v_1, x.u_1), \ldots, (x.v_l, x.u_l), model.only\}, & \text{if } x.ename =\gg \& y.aname \neq \gg \end{cases}$$

where $w_i \in V_M$ - is a violated attribute. In other words, such itemset consists of:

1. activity name, which is:
   - $y.aname$, if $y.aname \neq\gg$
   - or the activity name to which the event name is mapped, i.e. $em(x.ename)$ otherwise
2. all the event's attribute assignments
3. if the step is a violation, indicator of this violation:
   - a list of violated attributes $w_1, w_2, \ldots, w_m \in V_M$ if $(y.aname \neq \gg) \wedge (x.ename \neq \gg)$
   - or item $log.only$, if $(y.aname =\gg) \wedge (x.ename \neq \gg)$
   - or item $model.only$ if $(y.aname \neq \gg) \wedge (x.ename = \gg)$

Given a Data-Aware Replay Result DR, all Data-Aware Steps of which are contained in Z, a *transaction* is a subset of I that corresponds to a Data-Aware Step $z \in Z$.
Let D be a multiset of transactions over I corresponding to all Data-Aware Steps in Z, i.e. $D \in \mathcal{B}(2^I)$.

As an example let us consider alignment γ_2 from Table 2.2. The first step z_1 is not a violation, thus the corresponding transaction $T_{z_1} = \{a, (AR, 6000), (Ex, Pete)\}$, or, recalling our item names from I, $T_{z_1} = \{i_1, i_{14}, i_{10}\}$. The step z_2 is a move on log. Thus, the transaction corresponding to it is $T_{z_2} = \{c, (AR, 6000), (Ex, Pete), log.only\}$, or $T_{z_2} = \{i_3, i_{14}, i_{10}, i_{16}\}$. The step z_3 is a move with incorrect Ex; $T_{z_3} = \{b, (AR, 6000), (Ex, Ellen), Ex\}$, or $T_{z_3} = \{i_2, i_{14}, i_{11}, i_{16}\}$. The step z_4 is also a move with incorrect Ex; $T_{z_4} = \{d, (AR, 6000), (Ex, Mike), Ex\}$, or $T_{z_4} = \{i_4, i_{14}, i_{11}, i_{16}\}$. The step z_5 is a move on model; $T_{z_5} = \{e, (AR, 6000), (Ex, Mike), model.only\}$, or $T_{z_5} = \{i_5, i_{14}, i_{11}, i_{17}\}$. $T_{z_6} = \{g, (AR, 6000), (Ex, Pete)), $ or $T_{z_6} = \{i_7, i_{14}, i_{10}\}$.

Each subset of I has an associated statistical measure called *support*, denoted as *supp*. For an itemset $A \subseteq I$, $supp(A) = s$, if the fraction of transactions in D containing all items that are contained in A equals s. Please, note that **not** all possible subsets of I are included in D, only those corresponding to the Data-Aware Steps in Z.

For example, we consider a set $D = \{T_{z_1}, T_{z_2}, T_{z_3}, T_{z_4}, T_{z_5}, T_{z_6}, T\}$ of 6 transactions defined above and a transaction $T = \{i_{10}, i_{14}\}$, its support $supp(T) = \frac{4}{7}$, since 4 out of 7 transactions contain **both** items included in T.

An *association rule* is an implication of the form $A \Rightarrow B$, where A, B are subsets of I, $A, B \subseteq I$, and $A \cap B = \varnothing$. A is called the *antecedent* of the rule, and B is called the *consequent* of the rule.

Wu et al. in [29] proposed the *conditional-probability increment ratio (CPIR)* as the confidence measure of association rules. It can be utilized for quantifying the significance of association rules between violations, activities and attribute values. *CPIR* is a measure, aiming to quantify the rule's strength. Given $A, B \subseteq I, A \cap B = \varnothing$, $A, B \neq \varnothing$:

$$CPIR(A \Rightarrow B) = \frac{supp(A \cup B) - supp(A)supp(B)}{supp(A)(1 - supp(B))}$$

.

CPIR's value varies between -1 and 1, with negative values indicating negative relation. The closer the absolute value of the measure to 1, the stronger the relation and vice versa, the closer the value to 0, the weaker the relation.

As an example, we will calculate *CPIR* for the rule $\{i_{10}\} \Rightarrow \{i_{16}\}$ for our transactions in D, shown in the example above. $supp(\{i_{10}\}) = \frac{4}{7}$, $supp(\{i_{16}\}) = \frac{1}{7}$, $supp(\{i_{10}, i_{16}\}) = \frac{1}{7}$.

$$CPIR(\{i_{10}\} \Rightarrow \{i_{16}\}) = \frac{\frac{1}{7} - \frac{4}{7} * \frac{1}{7}}{\frac{4}{7} * (1 - \frac{1}{7})} = 0.125$$

As the *CPIR* value shows, the relation $\{i_{10}\} \Rightarrow \{i_{16}\}$ is quite weak.

2.3 Related Work

The approach proposed in this thesis can be decomposed into three basic parts: compliance checking and analysis, identifying relations between violation and underlying context information, and visualizing results for users without a strong technical background. In this section, we discuss literature related to each of the subjects.

Compliance Checking and Analysis

Our techniques are built upon violation data which is obtained during compliance checking. As stated in [25], there are two basic types of compliance checking: (1) *forward compliance checking* aims to design and implement processes where conformant behavior is enforced and (2) backward compliance checking aims to detect and localize non-conformant behavior. This thesis focuses on backward compliance checking based on event data.

In [5], four process perspectives are described: control-flow, organizational, case (also referred to as data) and time. The control-flow perspective considers existence relations between events (e.g. *if event A occurred, event B must also occur*) and order in which events occur (e.g. *event A must occur before event B*). The data, resource and time perspectives are the context attributes linked to each event. The resource perspective is tied to users or systems responsible for the occurrence of events and information about their department, positions, etc. The time perspective is related to the time of event occurrence. Finally, the data perspective considers all other context information, e.g. location, or purchased amount for procurement, or client ID, etc. Compliance requirements are often associated with multiple perspectives. Hence, we perform compliance analysis involving all of these perspectives.

Several techniques have been proposed for checking control-flow compliance and conformance. One of them is Declare-based checking, proposed in [20]. Similar to that, a LTL-based approach has been described in [6]. State-of-the-art techniques perform conformance checking by computing optimal alignments [9, 26] between traces in the event log and "best fitting" paths in the model. A compliance checking technique, aimed at providing diagnostic information for all deviations from compliant behavior, is proposed in [24]. This technique is based on the conformance checking techniques described in [9] and [11].

De Leoni et al. extend conformance checking to include also the data and resource perspectives in [19, 17]. The approaches employ checking not only the control-flow perspective, but also the data and resource perspectives of a process. This includes checking whether guards and conditions for execution paths employing attributes defined in the log are met. In [17] it is stated that a Petri Net with Data (defined in [18]) can be used to define the required perspectives.

In [12], modeling primitives for expressing time constraints between activities are proposed, which allow computing internal activity deadlines such that externally assigned deadlines are met. The technique presented in [25], focuses on backwards checking of temporal constraints in execution logs. It adopts the data-aware conformance approach described in [19] to check temporal rules. This allows expressing all temporal constraints discussed in previous works ([28, 21, 13, 10]) and above that, it allows checking cyclic temporal constraints.

For our approach, we employ a unified way of defining constraints using a Petri Net with Data. This allows to define and check control-flow rules as described in [24], data and resource rules as described in [18] and temporal constraints as described in [25].

Identifying Relations between a Violation and the Underlying Context Information

Our approach can be used to obtain compliance diagnostics. In order to provide diagnostic information, we need to discover relations between violations and the underlying context. The Data-Aware Replay Result described in Section 2.1.3 already has context information for all violations discovered. Nevertheless, examining each single step of each single alignment is enormously time-consuming for datasets larger than ten or a hand full of cases. Thus, context should be aggregated for each type of violation. However real processes are likely to contain a big number of violations with varying context. This creates (1) the need to measure the strength of the relationship between a violation and the underlying context, which is used for providing a general overview of compliance diagnostics. In order to obtain (2) the root cause of a specific problem, we need to find a way to discover what distinguishes violating steps from non-violating ones.

Both problems are solved using data-mining approaches. The approach proposed in [27] also uses decision trees to identify the root cause of problems, but it classifies process instances, while the approach in this thesis works with steps in alignments. Also, it does not employ any way of discovering association rules to measure the strength of the relationship between a violation and the underlying context. In this thesis, the approach proposed in [29] is utilized for this purpose.

Visualizing Results

Our aim in this thesis and the tools we have developed is to provide diagnostic information that is perceived easily by business users who are less familiar with technical knowledge. Hence visualization of diagnostics has been an important part of this work. Visualization in the context of process mining is not widely covered in literature, however there is a number of works worth mentioning.

In [16], the author proposes a technique to get insights into context-related information of event logs. Although, the proposed approach allows users to get useful context information, it does not focus on violating events or cases. Instead it aims to present the context data of an event log in general. Another paper involving visualization is [7] which, similarly to [15], focuses on visualizing the execution of event logs rather than the diagnostics.

Unlike the previously listed papers, [27] describes visualization of compliance diagnostics. In the mentioned paper, the implementation displays list of activities for each rule that can be drilled further down to context level. At the bottom level the user can observe various context data regarding the violation. In this thesis the implementation allows the user to explore not only violations related to a specific activity, but also violations related to specific values of context attributes. This lets user discover common context patterns across different violations and hence get better understanding of the observed problems.

Chapter 3

Obtaining Alignments to Identify Violations

In Chapter 1, we defined our research problem: *obtaining insights about the violations detected in an event log*. However, before we are able to provide such insights, we need to realize a way to detect violations. Violations are detected using a combination of compliance checking techniques for different process perspectives. The process of obtaining violation data is depicted in Figure 3.1.

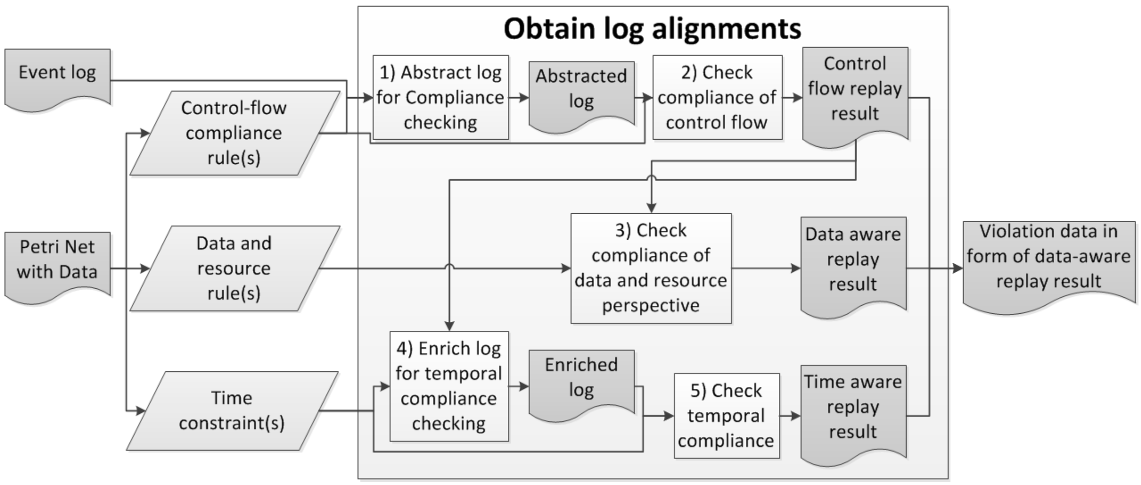

Obtain log alignments
Event log
Control-flow compliance rule(s)
Petri Net with Data
Data and resource rule(s)
Time constraint(s)
1) Abstract log for Compliance checking
Abstracted log
2) Check compliance of control flow
Control flow replay result
3) Check compliance of data and resource perspective
Data aware replay result
4) Enrich log for temporal compliance checking
Enriched log
5) Check temporal compliance
Time aware replay result
Violation data in form of data-aware replay result

Figure 3.1: Obtaining violation data

A process is decomposed into four perspectives: control-flow, data, organizational and time.

The *control-flow perspective* focuses on the ordering of activities and their presence or absence in process instances. If we recall our definition of log, provided in Section 2.1.1, this perspective is captured in event names and the sequence of events. Compliance rules restricting this perspective are captured in the specified behavior through the specified activity names and the specified sequence of activities.

The *organizational perspective* focuses on information about resources shown in the log, i.e., which actors (e.g., people, systems, roles, and departments) are involved and how they are related.

The *time perspective* is concerned with the timing of events. When events bear timestamps, it is possible to identify if an event met its deadline relative to some other event in the log.

The *data perspective* is concerned with all other information that is related to an event and may differ based on the context of the process. This perspective as well as the organizational and time perspectives are captured in event attributes and their values in the event log. The rules regarding this perspective are captured in *admissible values sets* for the attributes.

The input for obtaining log alignment (described in Figure 3.1) includes an event log and a Petri Net with Data. Such an event log is represented in the way described in Section 2.1.1. A Petri Net with Data is represented in the way described in Section 2.1.2. It contains information about control-flow, data, resource and time constraints regarding the analyzed process.

First of all, the event log along with the control-flow compliance rules serve as input for *abstracting the log for compliance checking.* This means replacing all event names, for which there are no corresponding activity names, with Ω. Thus all such events will be aligned to an activity named Ω. After that *control-flow rules* and the *abstracted log* are used as input for checking the control-flow compliance and obtain a *control-flow replay result.*

The obtained *control-flow replay result*, and data rules, are then used as an input for checking data and resource compliance and obtaining a corresponding *Data-Aware Replay Result.*

Independently from data and resource compliance, temporal compliance is checked as follows. First of all, the control-flow alignment, temporal constrains and abstracted log are utilized to obtain an *enriched log.* This log differs from the original *abstracted log* in a way that it has separate attributes with a timestamp for each event, for which a time constraint is defined. E.g. if we have a rule, restricting an event named *Goods Receipt* in the abstracted log, the enriched log would contain *_Goods_Receipt_Time* attribute bearing timestamp of the last occurred *Goods Receipt* event. Finally, temporal constraints are checked using the corresponding techniques and all three alignments are united into a resulting Data-Aware Replay Result.

The rest of this chapter is organized as follows. The example that is going to be used during the explanation is described in Section 3.1. Checking control-flow compliance is covered in Section 3.2. Data and resource compliance checking are described in Section 3.3 and temporal compliance checking is described in Section 3.4.

3.1 Motivating Example

Before describing how we detect violations, we introduce an example log and a rule specification. They will serve to illustrate the concepts throughout this thesis. The log is assumed to be recorded during execution of the process, described in Section 2.1.1. Let L be the analyzed event log.

$L = (\langle(\mathit{initiale\ request}, ((AR, 3000), (Ex, Pete), (TS, 0))),$
$(\mathit{check\ documents}, ((AR, 3000), (Ex, Pete), (TS, 1800))),$
$(\mathit{examine\ casually}, ((AR, 3000), (Ex, Ellen), (TS, 3600))),$
$(\mathit{decide}, ((AR, 3000), (Ex, Sara), (TS, 7200))),$
$(\mathit{pay\ compensation}, ((AR, 3000), (Ex, Pete), (TS, 10000)))\rangle,$

$\langle(\mathit{initiale\ request}, ((AR, 5000), (Ex, Pete), (TS, 0))),$
$(\mathit{examine\ casually}, ((AR, 5000), (Ex, Mike), (TS, 1000))),$
$(\mathit{check\ documents}, ((AR, 5000), (Ex, Sean), (TS, 2000))),$
$(\mathit{decide}, ((AR, 5000), (Ex, Mike), (TS, 2200))),$
$(\mathit{pay\ compensation}, ((AR, 5000), (Ex, Pete), (TS, 2600)))\rangle,$

$\langle(\mathit{initiale\ request}, ((AR, 6000), (Ex, Pete), (TS, 0))),$
$(\mathit{check\ documents}, ((AR, 6000), (Ex, Pete), (TS, 2000))),$
$(\mathit{examine\ thoroughly}, ((AR, 6000), (Ex, Sara), (TS, 7200))),$
$(\mathit{decide}, ((AR, 6000), (Ex, Sara), (TS, 8000))),$
$(\mathit{reject\ request}, ((AR, 6000), (Ex, Pete), (TS, 8200)))\rangle)$

The following attributes are included in the log:

- AR is a number, denoting the requested amount in Euros
- *Ex* is a person, responsible for the occurred event

- *TS* is a timestamp relative to start of the case in seconds

The specification is in terms of the Petri Net with Data depicted in Figure 3.2. The specification should satisfy the constraints described in [24]:

- Each pattern has a dedicated place *Initial* and a place *Final*
- A token in the final place defines the final marking of the pattern. When a pattern reaches its final marking, the pattern is properly completed (i.e., all other places of the net are empty).
- Every compliance pattern has a Pattern Instance corresponding to an instance of its compliance rule. The Pattern Instance starts as soon as an event occurs which triggers the Compliance Rule Instance. The Pattern Instance completes as soon as the condition of the Compliance Rule Instance is satisfied.
- The *I_st*-labeled transition in Petri-net pattern indicates the start of an instance of a control flow pattern (Pattern Instance) and the *I_cmp*-labeled transition in every pattern indicates the completion of an instance of the same control flow pattern.

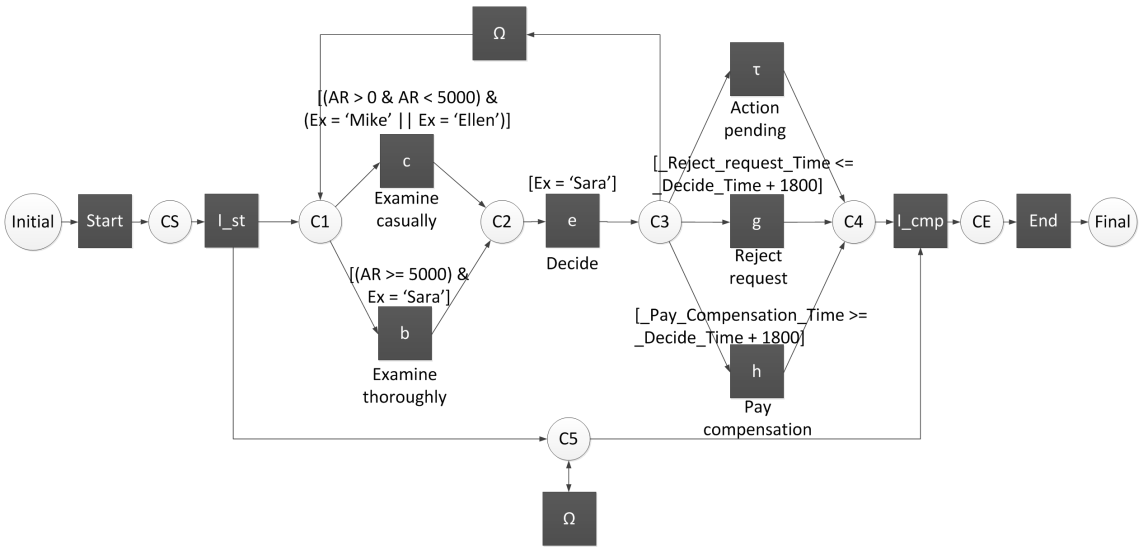

Ω
τ
Action pending
[(AR > 0 & AR < 5000) & (Ex = 'Mike' || Ex = 'Ellen')]
c
Examine casually
[_Reject_request_Time <= _Decide_Time + 1800]
[Ex = 'Sara']
Initial
Start
CS
I_st
C1
C2
e
Decide
C3
g
Reject request
C4
I_cmp
CE
End
Final
[(AR >= 5000) & Ex = 'Sara']
b
Examine thoroughly
[_Pay_Compensation_Time >= _Decide_Time + 1800]
h
Pay compensation
C5
Ω

Figure 3.2: Petri Net with Data

In this Petri Net with Data we use the following set of invisible transitions $T_{inv} = \{Start, I\_st, I\_cmp, \tau\}$. This Petri Net with Data expresses constraints regarding different perspectives.

For the **control-flow perspective**, the following constraints are expressed:

- Either activity *Examine casually* or *Examine thoroughly* should be executed, but not both;
- Activity *Decide* should eventually follow *Examine casually* or *Examine thoroughly*;
- Either of activities *Action pending, Reject request, or Pay compensation* should eventually be executed after *Decide*. Since activity *Action pending* is invisible, this rule would hold even if the other two are never performed;
- Sequence of one of the examining activities (either *Examine casually* or *Examine thoroughly*) and *Decide* activity can be repeated only after execution of some activity;
- All other activities execution is not restricted.

For the **data perspective**, the following constraints are expressed:

- Activity *Examine casually* can occur only when requested amount (AR) is below €5,000 and above €0;
- Activity *Examine thoroughly* can occur only when requested amount (AR) is above or equal €5,000.

For the **resource perspective**, the following constraints are expressed:

- Activity *Examine casually* can only be executed by Mike and Ellen;
- Activity *Examine thoroughly* can only be executed by Sara.

For the **time perspective**, the following constraints are expressed:

- If the request is rejected, the activity *Reject request* should be performed within 1800 seconds (i.e. 30 minutes) after activity *Decide*;
- If the request is approved, the activity *Pay compensation* should be performed after at least 1800 seconds (i.e. 30 minutes) after the activity *Decide*.

3.2 Control-flow Compliance Checking

For control-flow compliance checking we use the approach described in [25]. The first step is to abstract the log for compliance checking. During the abstraction, occurrences of any other events than the event(s) specified in the compliance rule are mapped onto the activity labeled Ω. For the sake of readability, we keep the log short and replace all event names with names of the transitions they correspond to. The result of abstracting our log is:

$L_{abstracted} = (\langle(\Omega, ((AR, 3000), (Ex, Pete), (TS, 0))),$
$(\Omega, ((AR, 3000), (Ex, Pete), (TS, 1800))),$
$(c, ((AR, 3000), (Ex, Ellen), (TS, 3600))),$
$(e, ((AR, 3000), (Ex, Sara), (TS, 7200))),$
$(h, ((AR, 3000), (Ex, Pete), (TS, 10000)))\rangle,$

$\langle(\Omega, ((AR, 5000), (Ex, Pete), (TS, 0))),$
$(c, ((AR, 5000), (Ex, Mike), (TS, 1000))),$
$(\Omega, ((AR, 5000), (Ex, Sean), (TS, 2000))),$
$(e, ((AR, 5000), (Ex, Mike), (TS, 2200))),$
$(h, ((AR, 5000), (Ex, Pete), (TS, 2600)))\rangle,$

$\langle(\Omega, ((AR, 6000), (Ex, Pete), (TS, 0))),$
$(\Omega, ((AR, 6000), (Ex, Pete), (TS, 2000))),$
$(b, ((AR, 6000), (Ex, Sara), (TS, 7200))),$
$(e, ((AR, 6000), (Ex, Sara), (TS, 8000))),$
$(g, ((AR, 6000), (Ex, Pete), (TS, 8200)))\rangle)$

After that, we can map events from the log to activities in the specification.
A **control-flow replay result** is a set of alignments (described in Section 2.1.3) with the difference that activities do not specify admissible values for any of the attributes, i.e. set of activity attributes $V_M = \varnothing$.

The control flow replay result for our example is shown in Table 3.1.

As mentioned in Section 2.1.3, events with name $\gg$ derive their attributes from the ones before them. When event with name $\gg$ occurs before any real event, there are no events to derive from, so all the attributes are derived from the first real event.

Table 3.1: Control flow replay result DR_{cf} of log $L_{abstracted}$

$\gamma_{\mathbf{cf1}}$		
z#	$\sigma_{\mathbf{L1}}$	$\sigma_{\mathbf{S1}}$
z_{11}	$\gg, ((AR, 3000), (Ex, Pete), (TS, 0))$	*Start*
z_{12}	$\gg, ((AR, 3000), (Ex, Pete), (TS, 0))$	*I_st*
z_{13}	$\Omega, ((AR, 3000), (Ex, Pete), (TS, 0))$	Ω
z_{14}	$\Omega, ((AR, 3000), (Ex, Pete), (TS, 1800))$	Ω
z_{15}	$c, ((AR, 3000), (Ex, Ellen), (TS, 3600))$	c
z_{16}	$e, ((AR, 3000), (Ex, Sara), (TS, 7200))$	e
z_{17}	$h, ((AR, 3000), (Ex, Pete), (TS, 10000))$	h
z_{18}	$\gg, ((AR, 3000), (Ex, Ellen), (TS, 10000))$	*I_cmp*
z_{19}	$\gg, ((AR, 3000), (Ex, Ellen), (TS, 10000))$	*End*
$\gamma_{\mathbf{cf2}}$		
z#	$\sigma_{\mathbf{L2}}$	$\sigma_{\mathbf{S2}}$
z_{21}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 0))$	*Start*
z_{22}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 0))$	*I_st*
z_{23}	$\Omega, ((AR, 5000), (Ex, Pete), (TS, 0))$	Ω
z_{24}	$c, ((AR, 5000), (Ex, Mike), (TS, 1000))$	c
z_{25}	$\Omega, ((AR, 5000), (Ex, Sean), (TS, 2000))$	Ω
z_{26}	$e, ((AR, 5000), (Ex, Mike), (TS, 2200))$	e
z_{27}	$h, ((AR, 5000), (Ex, Pete), (TS, 2600))$	h
z_{28}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 2600))$	*I_cmp*
z_{29}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 2600))$	*End*
$\gamma_{\mathbf{cf3}}$		
z#	$\sigma_{\mathbf{L3}}$	$\sigma_{\mathbf{S3}}$
z_{31}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 0))$	*Start*
z_{32}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 0))$	*I_st*
z_{33}	$\Omega, ((AR, 6000), (Ex, Pete), (TS, 0))$	Ω
z_{34}	$\Omega, ((AR, 6000), (Ex, Pete), (TS, 2000))$	Ω
z_{35}	$b, ((AR, 6000), (Ex, Sara), (TS, 7200))$	b
z_{36}	$e, ((AR, 6000), (Ex, Sara), (TS, 8000))$	e
z_{37}	$g, ((AR, 6000), (Ex, Pete), (TS, 8200))$	g
z_{38}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 8200))$	*I_cmp*
z_{39}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 8200))$	*End*

3.3 Data and Resource Compliance Checking

The approach described in this section is based on the one introduced in [19].

Input for this step is a *control-flow replay result* and data and resource constraints from a Petri Net with Data. As a result, a *Data-Aware Replay Result* of the form introduced in Section 2.1.3, can be computed. It is demonstrated in Table 3.2.

As shown in Table 3.2, alignment γ_{d2} contains a move with incorrect AR (See step z_{24}). There is also a move with incorrect Ex (See step z_{26}), i.e. there are both a data violation and a resource

Table 3.2: Data and resource alignment of log $L_{abstracted}$

	$\gamma_{\mathbf{d1}}$	
z#	$\sigma_{\mathbf{L1}}$	$\sigma_{\mathbf{S1}}$
z_{11}	$\gg, ((AR, 3000), (Ex, Pete), (TS, 0))$	$Start, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{12}	$\gg, ((AR, 3000), (Ex, Pete), (TS, 0))$	$I\_st, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{13}	$\Omega, ((AR, 3000), (Ex, Pete), (TS, 0))$	$\Omega, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{14}	$\Omega, ((AR, 3000), (Ex, Pete), (TS, 1800))$	$\Omega, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{15}	$c, ((AR, 3000), (Ex, Ellen), (TS, 3600))$	$c, ((AR, \{q \in \mathbb{Q}^+ \mid (q > 0) \wedge (q < 5000))), (Ex, \{Mike, Ellen\}))$
z_{16}	$e, ((AR, 3000), (Ex, Sara), (TS, 7200))$	$e, ((AR, \mathbb{Q}^+), (Ex, \{Sara\}))$
z_{17}	$h, ((AR, 3000), (Ex, Pete), (TS, 10000))$	$h, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{18}	$\gg, ((AR, 3000), (Ex, Ellen), (TS, 10000))$	$I\_cmp, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{19}	$\gg, ((AR, 3000), (Ex, Ellen), (TS, 10000))$	$End, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
	$\gamma_{\mathbf{d2}}$	
z#	$\sigma_{\mathbf{L2}}$	$\sigma_{\mathbf{S2}}$
z_{21}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 0))$	$Start, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{22}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 0))$	$I\_st, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{23}	$\Omega, ((AR, 5000), (Ex, Pete), (TS, 0))$	$\Omega, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{24}	$c, ((AR, 5000), (Ex, Mike), (TS, 1000))$	$c, ((AR, \{q \in \mathbb{Q}^+ \mid (q > 0) \wedge (q < 5000)\}), (Ex, \{Mike, Ellen\}))$
z_{25}	$\Omega, ((AR, 5000), (Ex, Sean), (TS, 2000))$	$\Omega, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{26}	$e, ((AR, 5000), (Ex, Mike), (TS, 2200))$	$e, ((AR, \mathbb{Q}^+), (Ex, \{Sara\}))$
z_{27}	$h, ((AR, 5000), (Ex, Pete), (TS, 2600))$	$h, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{28}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 2600))$	$I\_cmp, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{29}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 2600))$	$End, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
	$\gamma_{\mathbf{d3}}$	
z#	$\sigma_{\mathbf{L3}}$	$\sigma_{\mathbf{S3}}$
z_{31}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 0))$	$Start, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{32}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 0))$	$I\_st, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{33}	$\Omega, ((AR, 6000), (Ex, Pete), (TS, 0))$	$\Omega, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{34}	$\Omega, ((AR, 6000), (Ex, Pete), (TS, 2000))$	$\Omega, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{35}	$b, ((AR, 6000), (Ex, Sara), (TS, 7200))$	$b, ((AR, \{q \in \mathbb{Q}^+ \mid q >= 5000\}), (Ex, \{Sara\}))$
z_{36}	$e, ((AR, 6000), (Ex, Sara), (TS, 8000))$	$e((AR, \mathbb{Q}^+), (Ex, Sara))$
z_{37}	$g, ((AR, 6000), (Ex, Pete), (TS, 8200))$	$g, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{38}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 8200))$	$I\_cmp, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$
z_{39}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 8200))$	$End, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}))$

violation.

3.4 Temporal Compliance Checking

The approach described in this Section is based on the *Temporal Compliance Checking* approach introduced in [25]. In order to perform the checking, the following actions are taken. First, the abstracted event log is enriched with timing information. During this process, for all events, whose time is restricted by temporal constraints, special attributes bearing the value of their timestamps are created. Since all events are needed to have an equal set of attributes, these time attributes are added to every event. The name pattern of these attributes follows: *_[activity_name]_Time*, where *[activity_name]* is replaced with the actual name of the activity and all spaces ( ) in the activity name are replaced with _.

In our example, the following activities are considered by time restrictions: *Decide, Pay Compensation, Reject request.* Thus, the attributes, with which we enrich the log are: _Decide_Time, _Pay_Compensation_Time, and _Reject_request_Time. This results in the following enriched log for our example:

$L_{enriched} =$
$\langle\langle(\Omega, ((AR, 3000), (Ex, Pete), (TS, 0), (\_Decide\_Time, 0),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))),$
$(\Omega, ((AR, 3000), (Ex, Pete), (TS, 1800), (\_Decide\_Time, 0),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))),$
$(c, ((AR, 3000), (Ex, Ellen), (TS, 3600), (\_Decide\_Time, 0),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))),$
$(e, ((AR, 3000), (Ex, Sara), (TS, 7200), (\_Decide\_Time, 7200),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))),$
$(h, ((AR, 3000), (Ex, Pete), (TS, 10000), (\_Decide\_Time, 7200),$
$(\_Pay\_Compensation\_Time, 10000), (\_Reject\_request\_Time, 0)))\rangle,$

$\langle(\Omega, ((AR, 5000), (Ex, Pete), (TS, 0), (\_Decide\_Time, 0),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))),$
$(c, ((AR, 5000), (Ex, Mike), (TS, 1000), (\_Decide\_Time, 0),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))),$
$(\Omega, ((AR, 5000), (Ex, Sean), (TS, 2000), (\_Decide\_Time, 0),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))),$
$(e, ((AR, 5000), (Ex, Mike), (TS, 2200), (\_Decide\_Time, 2200),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))),$
$(h, ((AR, 5000), (Ex, Pete), (TS, 2600), (\_Decide\_Time, 2200),$
$(\_Pay\_Compensation\_Time, 2600), (\_Reject\_request\_Time, 0)))\rangle,$

$\langle(\Omega, ((AR, 6000), (Ex, Pete), (TS, 0), (\_Decide\_Time, 0),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))),$
$(\Omega, ((AR, 6000), (Ex, Pete), (TS, 2000), (\_Decide\_Time, 0),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))),$
$(b, ((AR, 6000), (Ex, Ellen), (TS, 7200), (\_Decide\_Time, 0),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))),$
$(e, ((AR, 6000), (Ex, Sara), (TS, 8000), (\_Decide\_Time, 8000),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))),$
$(g, ((AR, 6000), (Ex, Pete), (TS, 8200), (\_Decide\_Time, 8000),$
$(\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 8200)))\rangle\rangle$

Second, control-flow replay result is enriched with those attributes for log events and restrictions based on actual attribute values for specification activities. The resulting temporal alignment is shown in Table 3.3.

Table 3.3: Temporal alignment of log $L_{enriched}$

$\gamma_{\mathbf{t1}}$		
z#	$\sigma_{\mathbf{L1}}$	$\sigma_{\mathbf{S1}}$
z_{11}	$\gg, ((AR, 3000), (Ex, Pete), (TS, 0),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$Start, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{12}	$\gg, ((AR, 3000), (Ex, Pete), (TS, 0),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$I\_st, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{13}	$\Omega, ((AR, 3000), (Ex, Pete), (TS, 0),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$\Omega, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{14}	$\Omega, ((AR, 3000), (Ex, Pete), (TS, 1800),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$\Omega, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{15}	$c, ((AR, 3000), (Ex, Ellen), (TS, 3600),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$c, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{16}	$e, ((AR, 3000), (Ex, Sara), (TS, 7200),$ $(\_Decide\_Time, 7200),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$e, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{17}	$h, ((AR, 3000), (Ex, Pete), (TS, 10000),$ $(\_Decide\_Time, 7200),$ $(\_Pay\_Compensation\_Time, 10000),$ $(\_Reject\_request\_Time, 0))$	$h, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time,$ $\{q \in \mathbb{Q}^{+} \mid q >= 9000\}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{18}	$\gg, ((AR, 3000), (Ex, Pete), (TS, 10000),$ $(\_Decide\_Time, 7200),$ $(\_Pay\_Compensation\_Time, 10000),$ $(\_Reject\_request\_Time, 0))$	$I\_cmp, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{19}	$\gg, ((AR, 3000), (Ex, Pete), (TS, 10000),$ $(\_Decide\_Time, 7200),$ $(\_Pay\_Compensation\_Time, 10000),$ $(\_Reject\_request\_Time, 0))$	$End, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
$\gamma_{\mathbf{t2}}$		
z#	$\sigma_{\mathbf{L2}}$	$\sigma_{\mathbf{S2}}$
z_{21}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 0),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$Start, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{22}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 0),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$I\_st, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$

z_{23}	$\Omega, ((AR, 5000), (Ex, Pete), (TS, 0), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$\Omega, ((\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{24}	$c, ((AR, 5000), (Ex, Mike), (TS, 1000), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$c, ((\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{25}	$\Omega, ((AR, 5000), (Ex, Sean), (TS, 2000), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$\Omega, ((\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{26}	$e, ((AR, 5000), (Ex, Mike), (TS, 2200), (\_Decide\_Time, 2200), (\_Pay\_Compensation\_Time, 2600), (\_Reject\_request\_Time, 0))$	$e, ((\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{27}	$h, ((AR, 5000), (Ex, Pete), (TS, 2600), (\_Decide\_Time, 2200), (\_Pay\_Compensation\_Time, 2600), (\_Reject\_request\_Time, 0))$	$h, ((\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \{q \in \mathbb{Q}^+ \mid q >= 4000\}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{28}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 2600), (\_Decide\_Time, 2200), (\_Pay\_Compensation\_Time, 2600), (\_Reject\_request\_Time, 0))$	$I\_cmp, ((\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{29}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 2600), (\_Decide\_Time, 2200), (\_Pay\_Compensation\_Time, 2600), (\_Reject\_request\_Time, 0))$	$End, ((\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
	γ_{t3}	
z#	$\sigma_{\mathbf{L3}}$	$\sigma_{\mathbf{S3}}$
z_{31}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 0), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$Start, ((\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{32}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 0), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$I\_st, ((\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{33}	$\Omega, ((AR, 6000), (Ex, Pete), (TS, 0), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$\Omega, ((\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{34}	$\Omega, ((AR, 6000), (Ex, Pete), (TS, 2000), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$\Omega, ((\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{35}	$b, ((AR, 6000), (Ex, Sara), (TS, 7200), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$b, ((\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$

z_{36}	$e, ((AR, 6000), (Ex, Sara), (TS, 8000),$ $(\_Decide\_Time, 8000),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$e, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{37}	$g, ((AR, 6000), (Ex, Pete), (TS, 8200),$ $(\_Decide\_Time, 8000),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 8200))$	$g, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time,$ $\{q \in \mathbb{Q}^+ \mid q <= 9800\}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{38}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 8200),$ $(\_Decide\_Time, 7200),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 8200))$	$I\_cmp, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{39}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 8200),$ $(\_Decide\_Time, 7200),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 8200))$	$End, ((\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$

As can be seen in the table, there is a single violation of a temporal rule: step z_{27} in alignment γ_{t2} is performed with incorrect time.

Finally, after obtaining alignments for all perspectives, the resulting alignment is compiled. It is composed of merged information from all other perspectives. The result is shown in the Table 3.4.

Table 3.4: Resulting Data-Aware Replay Result of log $L_{enriched}$

γ_1		
z#	σ_{L1}	σ_{S1}
z_{11}	$\gg, ((AR, 3000), (Ex, Pete), (TS, 0),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$Start, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{12}	$\gg, ((AR, 3000), (Ex, Pete), (TS, 0),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$I\_st, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{13}	$\Omega, ((AR, 3000), (Ex, Pete), (TS, 0),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$\Omega, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{14}	$\Omega, ((AR, 3000), (Ex, Pete), (TS, 1800),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$\Omega, ((AR, \mathbb{Q}^+), (Ex, \mathbb{U}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{15}	$c, ((AR, 3000), (Ex, Ellen), (TS, 3600),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$c, ((AR, \{q \in \mathbb{Q}^+ \mid (q > 0) \wedge (q < 5000))),$ $(Ex, \{Mike, Ellen\}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{16}	$e, ((AR, 3000), (Ex, Sara), (TS, 7200),$ $(\_Decide\_Time, 7200),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$e, ((AR, \mathbb{Q}^+), (Ex, \{Sara\}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$

z_{17}	$h, ((AR, 3000), (Ex, Pete), (TS, 10000), (\_Decide\_Time, 7200), (\_Pay\_Compensation\_Time, 10000), (\_Reject\_request\_Time, 0))$	$h, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}), (\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \{q \in \mathbb{Q}^{+} \mid q >= 9000\}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{18}	$\gg, ((AR, 3000), (Ex, Pete), (TS, 10000), (\_Decide\_Time, 7200), (\_Pay\_Compensation\_Time, 10000), (\_Reject\_request\_Time, 0))$	$I\_cmp, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}), (\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{19}	$\gg, ((AR, 3000), (Ex, Pete), (TS, 10000), (\_Decide\_Time, 7200), (\_Pay\_Compensation\_Time, 10000), (\_Reject\_request\_Time, 0))$	$End, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}), (\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
	$\boldsymbol{\gamma_2}$	
z#	$\boldsymbol{\sigma_{L2}}$	$\boldsymbol{\sigma_{S2}}$
z_{21}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 0), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$Start, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}), (\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{22}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 0), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$I\_st, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}), (\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{23}	$\Omega, ((AR, 5000), (Ex, Pete), (TS, 0), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$\Omega, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}), (\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{24}	$c, ((AR, 5000), (Ex, Mike), (TS, 1000), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$c, ((AR, \{q \in \mathbb{Q}^{+} \mid (q > 0) \wedge (q < 5000)\})), (Ex, \{Mike, Ellen\}), (\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{25}	$\Omega, ((AR, 5000), (Ex, Sean), (TS, 2000), (\_Decide\_Time, 0), (\_Pay\_Compensation\_Time, 0), (\_Reject\_request\_Time, 0))$	$\Omega, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}), (\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{26}	$e, ((AR, 5000), (Ex, Mike), (TS, 2200), (\_Decide\_Time, 2200), (\_Pay\_Compensation\_Time, 2600), (\_Reject\_request\_Time, 0))$	$e, ((AR, \mathbb{Q}^{+}), (Ex, \{Sara\}), (\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{27}	$h, ((AR, 5000), (Ex, Pete), (TS, 2600), (\_Decide\_Time, 2200), (\_Pay\_Compensation\_Time, 2600), (\_Reject\_request\_Time, 0))$	$h, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}), (\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \{q \in \mathbb{Q}^{+} \mid q >= 4000\}), (\_Reject\_request\_Time, \mathbb{Q}))$
z_{28}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 2600), (\_Decide\_Time, 2200), (\_Pay\_Compensation\_Time, 2600), (\_Reject\_request\_Time, 0))$	$I\_cmp, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}), (\_Decide\_Time, \mathbb{Q}), (\_Pay\_Compensation\_Time, \mathbb{Q}), (\_Reject\_request\_Time, \mathbb{Q}))$

z_{29}	$\gg, ((AR, 5000), (Ex, Pete), (TS, 2600),$ $(\_Decide\_Time, 2200),$ $(\_Pay\_Compensation\_Time, 2600),$ $(\_Reject\_request\_Time, 0))$	$End, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
	$\gamma_{\mathbf{3}}$	
z#	$\sigma_{\mathbf{L3}}$	$\sigma_{\mathbf{S3}}$
z_{31}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 0),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$Start, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{32}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 0),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$I\_st, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{33}	$\Omega, ((AR, 6000), (Ex, Pete), (TS, 0),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$\Omega, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{34}	$\Omega, ((AR, 6000), (Ex, Pete), (TS, 2000),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$\Omega, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{35}	$b, ((AR, 6000), (Ex, Sara), (TS, 7200),$ $(\_Decide\_Time, 0),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$b, ((AR, \{q \in \mathbb{Q}^{+} \mid q >= 5000\}),$ $(Ex, \{Sara\}), (\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{36}	$e, ((AR, 6000), (Ex, Sara), (TS, 8000),$ $(\_Decide\_Time, 8000),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 0))$	$e, ((AR, \mathbb{Q}^{+}), (Ex, Sara),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{37}	$g, ((AR, 6000), (Ex, Pete), (TS, 8200),$ $(\_Decide\_Time, 8000),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 8200))$	$g, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time,$ $\{q \in \mathbb{Q}^{+} \mid q <= 9800\}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{38}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 8200),$ $(\_Decide\_Time, 7200),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 8200))$	$I\_cmp, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$
z_{39}	$\gg, ((AR, 6000), (Ex, Pete), (TS, 8200),$ $(\_Decide\_Time, 7200),$ $(\_Pay\_Compensation\_Time, 0),$ $(\_Reject\_request\_Time, 8200))$	$End, ((AR, \mathbb{Q}^{+}), (Ex, \mathbb{U}),$ $(\_Decide\_Time, \mathbb{Q}),$ $(\_Pay\_Compensation\_Time, \mathbb{Q}),$ $(\_Reject\_request\_Time, \mathbb{Q}))$

In this way a Data-Aware Replay Result containing information about violations from all process perspectives is obtained. The replay result will be analyzed in the way described in the following chapters.

Chapter 4

Identifying and Ranking Problems

In Chapter 3, we described how we obtain Data-Aware Replay Result containing the violation data. In this chapter, we describe how we identify problems in the given Data-Aware Replay Result. This results include violation statistics and a ranked list of problems.

As stated in Section 2.1.3, Data-Aware Replay Result consists of alignments. Each alignment corresponds to a single case. Thus, in order to understand what problems are present in the process, a user would have to examine the alignments for an entire log. Moreover, in order to get the idea of the context corresponding to each problem, it is required to study every single step of the alignments and manually relate the violating steps to the underlying context.

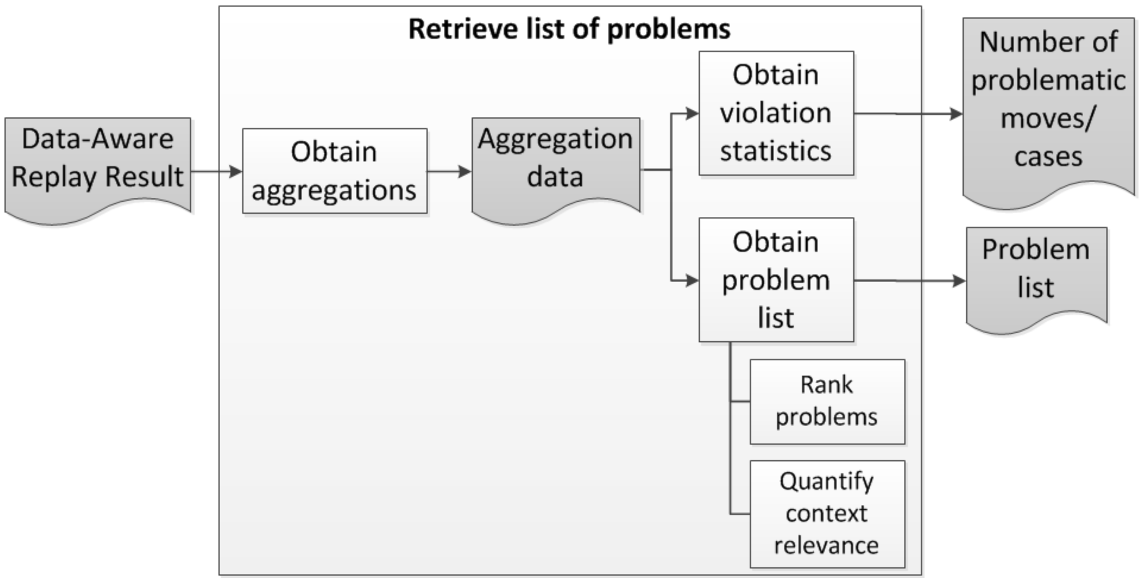

Retrieve list of problems
Data-Aware Replay Result
Obtain aggregations
Aggregation data
Obtain violation statistics
Obtain problem list
Rank problems
Quantify context relevance
Number of problematic moves/ cases
Problem list

Figure 4.1: Approach step 1 details

This phase of the approach is depicted in Figure 4.1. It is aimed at obtaining the general overview of the violations present in the process and the underlying diagnostics. The input is a result of compliance checking in the form of *Data-Aware Replay Result*. The Replay Result is utilized to obtain *aggregations*, which will be explained in Section 4.1. The aggregations serve as the input for obtaining *violation statistics* and a *list of problems*. Each item of the list has an assigned relevance metric and a rank.

The statistics include number of violations of each kind for each attribute value and activity. They are described in Section 4.2.

The list of problems is shown to the user as a report containing textual description of discovered problems. Each item of the list has an assigned relevance value and a rank value. Using the rank value, the list is sorted such that most important or severe violations appear at the top. Using the relevance value, the list can be refined in order to keep only the most relevant violations for each specific activity, or the most relevant context for each specific violation. The structure of the list and the way the metrics are computed are described in Section 4.3.

4.1 Obtaining Aggregations

The aggregations consist of problematic subsets. Basically, each problematic subset is a subset of the multiset of Data-Aware Steps Z, where each step represents a violation and has one or more specific features, be it bearing a specific activity name, or an attribute value or representing a specific kind of violation. These subsets allow us to relate the discovered violations to the underlying context.

4.1.1 Aggregating Multisets

In this section we describe which multisets of Data-aware-steps are chosen for aggregation.

We assume the given Data-Aware Replay Result DR in the form, defined in Section 2.1.3, all Data-Aware Steps of which are contained in Z. In order to define the multisets, we use functions defined in Section 2.1.4. Table 4.1 defines the multisets and shows examples for each subset, using the example Data-Aware Replay Result described in Table 3.4.

Table 4.1: Aggregating multisets

Multiset definition	Input parameters	Description	Example
$R_{name}(aname) = \{z \in Z \mid sname(z) = aname\}$	activity name $aname \in A$	a subset of all Data-Aware Steps with the given activity name *aname*	$R_{name}(c) = \{z_{15}, z_{24}\}$
$R_{val}(v, u) = \{z \in Z \mid val(z, v) = u\}$	attribute $v \in V_L$, its value $u \in G_v$	a subset of all Data-Aware Steps with the provided attribute value $u \in G_v$ of the given attribute $v \in V_L$.	$R_{val}(Ex, Sara) = \{z_{16}, z_{35}, z_{36}\}$
$R_{model.only} = \{z \in Z \mid P_{model.only}(z)\}$	-	a subset of all Data-Aware Steps, which are moves on model.	$R_{model.only} = \varnothing$
$R_{log.only} = \{z \in Z \mid P_{log.only}(z)\}$	-	a subset of all Data-Aware Steps, which are moves on log.	$R_{log.only} = \varnothing$
$R_{incorrect.attr}(w) = \{z \in Z \mid P_{wrong.v}(z, v)\}$	specification attribute $w \in V_M$	a subset of all Data-Aware Steps, which are moves with incorrect w.	$R_{incorrect.attr}(Ex) = \{z_{26}\}$
$R_{incorrect.data} = \{z \in Z \mid \exists\, w[w \in V_M : P_{incorrect.attr}(z, w)]\}$	-	a subset of all Data-Aware Steps, which are moves with incorrect attribute for any of the attributes.	$R_{incorrect.data} = \{z_{24}, z_{26}, z_{27}\}$
$R_{violation} = R_{model.only} \cup R_{log.only} \cup R_{incorrect.data}$	-	a subset of all Data-Aware Steps, which have any kind of violation.	$R_{violation} = \{z_{24}, z_{26}, z_{27}\}$
$S_{model.only}(aname) = R_{name}(aname) \cap R_{model.only}$	activity name $aname \in A$	a subset of all Data-Aware Steps, which have given activity name and are *moves on model.*	$S_{model.only}(c) = \varnothing$

Table 4.1: Aggregating multisets (continued)

Multiset definition	Input parameters	Description	Example
$S_{log.only}(aname) = R_{name}(aname) \cap R_{log.only}$	activity name $aname \in A$	a subset of all Data-Aware Steps, which have given activity name and are *moves on log.*	$S_{log.only}(c) = \varnothing$
$S_{dataviolation}(aname, w) = \{z \in Z \mid sname(z) = aname \wedge P_{incorrect.attr}(z, w)\}$	activity name $aname \in A$, specification attribute $w \in V_M$	a subset of all Data-Aware Steps, which have given activity name and have *incorrect value* of given attribute w.	$S_{dataviolation}(c, AR) = \varnothing$ $S_{dataviolation}(c, Ex) = \{z_{24}\}$
$S_{anyviolation}(aname) = R_{name}(aname) \cap R_{violation}$	activity name $aname \in A$	a subset of all Data-Aware Steps, which have given activity name and bear any kind of violation.	$S_{anyviolation}(h) = \{z_{27}\}$
$S_{assignment}(v, u) = S_{val}(v, u) \cap R_{violation}$	attribute $v \in V_L$, its value $u \in G_v$	a subset of all Data-Aware Steps, which have the given value u for the given attribute v and bear any kind of violation	$S_{assignment}(AR, 5000) = \{z_{24}, z_{26}, z_{27}\}$
$S_{model.only.assignment}(aname, v, u) = S_{model.only}(aname) \cap S_{assignment}(v, u)$	activity name $aname \in A$, attribute $v \in V_L$, its value $u \in G_v$	a subset of all Data-Aware Steps, which have the given activity name *aname*, the given value u of the given attribute v and are *moves on model*	$S_{model.only.assignment}(c, AR, 3000) = \varnothing$

Table 4.1: Aggregating multisets (continued)

Multiset definition	Input parameters	Description	Example
$S_{assignment.model.only}(aname, v, u) = S_{assignment}(v, u) \cap S_{model.only}(aname)$	activity name $aname \in A$, attribute $v \in V_L$, its value $u \in G_v$	the set is the same as the previous one, but list items to which they correspond bear different meanings. The list items are introduced in Section 4.3.1.	$S_{assignment.model.only}(c, AR, 3000) = \varnothing$
$S_{log.only.assignment}(aname, v, u) = S_{log.only}(aname) \cap S_{assignment}(v, u)$	activity name $aname \in A$, attribute $v \in V_L$, its value $u \in G_v$	a subset of all Data-Aware Steps, which have the given activity name *aname*, the given value u of the given attribute v and are *moves on log*	$S_{log.only.assignment}(c, AR, 3000) = \varnothing$
$S_{assignment.log.only}(aname, v, u) = S_{assignment}(v, u) \cap S_{log.only}(aname)$	activity name $aname \in A$, attribute $v \in V_L$, its value $u \in G_v$	the set is the same as the previous one, but list items to which they correspond bear different meanings. The list items are introduced in Section 4.3.1.	$S_{assignment.log.only}(c, AR, 3000) = \varnothing$
$S_{dataviolation.assignment}(aname, v, u, w) = S_{dataviolation}(aname, w) \cap S_{assignment}(v, u)$	activity name *aname*, attribute $v \in V_L$, its value $u \in G_v$, specification attribute $w \in V_M$	a subset of all Data-Aware Steps, which have the given activity name *aname*, the given value u of the given attribute v and are moves with incorrect attribute w	$S_{dataviolation.assignment}(c, AR, 3000, AR) = \varnothing$

Table 4.1: Aggregating multisets (continued)

Multiset definition	Input parameters	Description	Example
$S_{assignment.dataviolation}(aname, v, u, w) = S_{assignment}(v, u) \cap S_{dataviolation}(aname, w)$	activity name *aname*, attribute $v \in V_L$, its value $u \in G_v$, specification attribute $w \in V_M$	the set is the same as the previous one, but list items to which they correspond bear different meanings. The list items are introduced in Section 4.3.1.	$S_{assignment.dataviolation}(c, AR, 5000, AR) = \{z_{24}\}$

4.2 Obtaining Violation Statistics

Violation statistics allow to get impression of the severity of the violations present in the analyzed process. The idea is to show number of violations of each kind for each element of the Data-Aware Replay Result (activity names and attribute values). This includes number of Data-Aware Steps with violation and number of cases with violation.

For each activity $aname \in A$ the statistics include the following:

- Number of moves on log (cardinality of multiset $S_{log.only}(aname)$)
- Number of moves on model ($| S_{model.only}(aname) |$)
- For each $w \in V_M$, number of moves with **incorrect** w ($| S_{dataviolation}(aname, w) |$)
- Number of synchronous moves ($| R_{name}(aname) | - | S_{model.only}(aname) \cup S_{log.only}(aname) |$)
- For each $w \in V_M$, number of moves with **correct** w ($| R_{name}(aname) | - | S_{dataviolation}(aname, w) |$)

Similarly, for each attribute $v \in V_L^{context}$ and its value $u \in G_v$ the statistics include the following:

- Number of moves on log ($| R_{val}(v, u) \cap R_{log.only} |$)
- Number of moves on model ($| R_{val}(v, u) \cap R_{model.only} |$)
- For each $w \in V_M$, number of moves with incorrect w ($| R_{val}(v, u) \cap R_{incorrect.attr}(w) |$)
- Number of synchronous moves ($| R_{val}(v, u) | - | R_{val}(v, u) \cap (R_{log.only} \cup R_{model.only}) |$)
- For each $w \in V_M$, number of moves with **correct** w ($| R_{val}(v, u) | - | R_{val}(v, u) \cap R_{incorrect.attr}(w) |$)

Please note that the statistics are not produced for the time attributes. The reason is that cases might occur within huge timespan and timestamps might be incomparable and including them might bias the report. This restriction only considers the default timestamp and the attributes with which the log is enriched during temporal compliance checking.

To illustrate the obtained result we show the statistics obtained from the example Data-Aware Replay Result in Table 4.2. The table rows correspond to activities, while similar figures can be provided for values of any attribute. Please note that if an attribute is numeric, the statistics are presented not for each single value, but for value intervals. Since in the demonstrated example the numeric attribute AR has only three different values, it is not split into intervals.

Table 4.2: Violation statistics L

Activity name	**Moves on log**	**Moves on model**	**Incorrect** *Ex*	**Incorrect** *Ar*	**Incorrect** *_Decide_Time*	**Incorrect** *_Pay_Compansation_Time*	**Incorrect** *_Reject_request_Time*
b	0	0	0	0	0	0	0
c	0	0	0	1	0	0	0
e	0	0	1	0	0	0	0
h	0	0	0	0	0	0	1
g	0	0	0	0	0	0	0
Ω	0	0	0	0	0	0	0
Start	0	0	0	0	0	0	0
End	0	0	0	0	0	0	0
I_st	0	0	0	0	0	0	0
I_cmp	0	0	0	0	0	0	0

Activity name	**Synchronous moves**	**Correct** *Ex*	**Correct** *Ar*	**Correct** *_Decide_Time*	**Correct** *_Pay_Compansation_Time*	**Correct** *_Reject_request_Time*
b	1	1	1	1	1	1
c	2	2	1	2	2	2
e	3	2	3	3	3	3
h	2	2	2	2	2	1
g	1	1	1	1	1	1
Ω	6	6	6	6	6	6
Start	3	3	3	3	3	3
End	3	3	3	3	3	3
I_st	3	3	3	3	3	3
I_cmp	3	3	3	3	3	3

4.3 Obtaining Problem List

In order to obtain a problem list, the approach requires several inputs. The inputs are: aggregated multisets, importance function (explained in Section 4.3.2) and relevance thresholds (explained in Section 4.3.3) for including items into the list. The aggregations are obtained from Data-Aware Replay Result as it is described above, while the other inputs are provided by the user. The inputs are summarized in Figure 4.2.

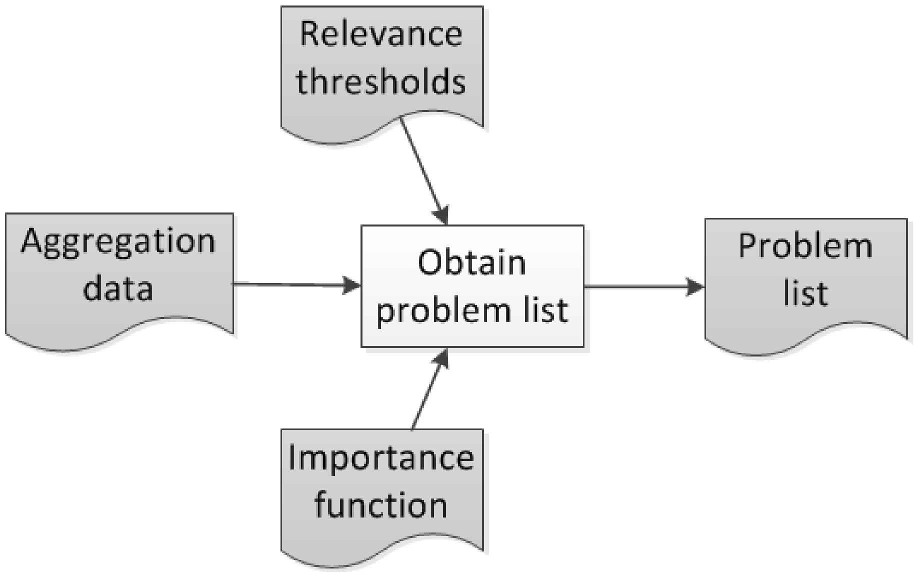


Figure 4.2: Obtaining list of problems inputs

4.3.1 Problem List

Let PL be a problem list. PL is a tuple, $PL = (pli_{aname_1}, \dots, pli_{aname_n}, pli_{(v_1, u_1^{v_1})}, \dots, pli_{(v_m, u_k^{v_m})})$, where:

- $aname_i \in A$ for $i = 1, \dots, \mid A \mid$ is an activity name;
- $v_i \in V_L^{context}$ for $i = 1, \dots, \mid V_L^{context} \mid$ is an attribute;
- $u_j^{v_i} \in G_{v_i}$ for $i = 1, \dots, \mid V_L^{context} \mid, j = 1, \dots, \mid G_{v_i} \mid$ is an attribute value

As shown, there are two types of items in the list: pli_{aname} and $pli_{(v,u)}$. Both of them are **tuples**, consisting of a textual description *desc*, and a number of sub-items.

The list items, their textual descriptions, and hierarchy are described in Table 4.3. Each item corresponds to a specific activity name, attribute assignment, and a violation type. These determine to which of the problematic multisets defined in Table 4.1 the item corresponds. The triangular brackets in the textual description of an item means that their content should be replaced with an actual name of an activity, or an attribute or its value.

Table 4.3: Problem list structure

Item level	Item definition	Textual representation	Corresponding multiset
1	$pli_{aname} = (desc,$ $pli^{aname}_{model.only}, pli^{aname}_{log.only},$ $pli^{aname}_{wrong.w_1}, \ldots, pli^{aname}_{wrong.w_m})$	$<aname>$	$S_{anyviolation}(aname)$
2	$pli^{aname}_{model.only} = (desc,$ $pli^{aname,(v_1,u_1^{v1})}_{model.only}, \ldots,$ $pli^{aname,(v_m,u_k^{vm})}_{model.only})$	Activity $<aname>$is skipped	$S_{model.only}(aname)$
3	$pli^{aname,(v,u)}_{model.only} = (desc)$	Activity $<aname>$is skipped, when $<v>= <u>$	$S_{model.only.assignment}(aname, v, u)$
2	$pli^{aname}_{log.only} = (desc,$ $pli^{aname,(v_1,u_1^{v1})}_{log.only}, \ldots,$ $pli^{aname,(v_m,u_k^{vm})}_{log.only})$	Activity $<aname>$is performed where it shouldn't	$S_{logdel.only}(aname)$
3	$pli^{aname,(v,u)}_{log.only} = (desc)$	Activity $<aname>$is performed where it shouldn't, when $<v>= <u>$	$S_{log.only.assignment}(aname, v, u)$
2	$pli^{aname}_{wrong.w} = (desc,$ $pli^{aname,(v_1,u_1^{v1})}_{wrong.w}, \ldots,$ $pli^{aname,(v_m,u_k^{vm})}_{wrong.w})$	Activity $<aname>$is performed with incorrect	$S_{dataviolation}(aname, w)$
3	$pli^{aname,(v,u)}_{wrong.w} = (desc)$	Activity $<aname>$is performed with incorrect $<w>$, when $<v>= <u>$	$S_{dataviolation.assignment}(aname, v, u, w)$

Table 4.3: Problem list structure (continued)

Item level	Item definition	Textual representation	Corresponding multiset
1	$pli_{(v,u)} = (desc,$ $pli^{(v,u),aname}_{model.only},$ $pli^{(v,u),aname}_{log.only},$ $pli^{(v,u),aname}_{wrong.w_1}, \dots,$ $pli^{(v,u),aname}_{wrong.w_m})$	When <v>= <u>	$S_{assignment}(v, u)$
2	$pli^{(v,u),aname}_{model.only} = (desc)$	When <v>= <u>, activity <$aname$>is skipped	$S_{assignment.model.only}(aname, v, u)$
2	$pli^{(v,u),aname}_{log.only} = (desc)$	When <v>= <u>activity, <$aname$>is performed where it shouldn't	$S_{assignment.log.only}(aname, v, u)$
2	$pli^{(v,u),aname}_{wrong.w} = (desc)$	When <v>= <u>, activity <$aname$>is performed with incorrect <w>	$S_{assignment.dataviolation}(aname, v, u, w)$

In order to illustrate our approach, we demonstrate the example list of problems, obtained from the example Data-Aware Replay Result shown in Table 3.4.

The example shows textual description of every item and lists its sub-items as a nested list.

- $pli_{(AR,5000)}$: When AR = *5000*;
  - $pli^{(AR,5000),c}_{wrong.AR}$: When AR = *5000*, activity c is performed with incorrect AR;
  - $pli^{(AR,5000),e}_{wrong.Ex}$: When AR = *5000*, activity e is performed with incorrect Ex;
  - $pli^{(AR,5000),h}_{wrong.Time}$: When AR = *5000*, activity h is performed with incorrect $Time$;
- pli_c: c
  - $pli^{c}_{wrong.AR}$: Activity c is performed with incorrect AR;
    - $pli^{c,(Ex,Mike)}_{wrong.AR}$: Activity c is performed with incorrect AR, when Ex = *Mike*;
    - $pli^{c,(AR,5000)}_{wrong.AR}$: Activity c is performed with incorrect AR, when AR = *5000*;
- pli_e: e
  - $pli^{e}_{wrong.Ex}$: Activity e is performed with incorrect Ex;
    - $pli^{e,(Ex,Mike)}_{wrong.Ex}$: Activity e is performed with incorrect Ex, when Ex = *Mike*;
    - $pli^{e,(AR,5000)}_{wrong.Ex}$: Activity e is performed with incorrect Ex, when AR = *5000*;
- pli_h: h
  - $pli^{h}_{wrong.Time}$: Activity h is performed with incorrect $Time$;
    - $pli^{h,(Ex,Pete)}_{wrong.Time}$: Activity h is performed with incorrect $Time$, when Ex = *Pete*;
    - $pli^{h,(AR,5000)}_{wrong.Time}$: Activity h is performed with incorrect $Time$, when AR = *5000*;
- $pli_{(Ex,Mike)}$: When Ex = *Mike*;
  - $pli^{(Ex,Mike),c}_{wrong.AR}$: When Ex = *Mike*, activity c is performed with incorrect AR;
  - $pli^{(Ex,Mike),e}_{wrong.Ex}$: When Ex = *Mike*, activity e is performed with incorrect Ex;
- $pli_{(Ex,Pete)}$: When Ex = *Pete*;
  - $pli^{(Ex,Pete),h}_{wrong.Time}$: When Ex = *Pete*, activity h is performed with incorrect $Time$;

4.3.2 Sorting List

The list items are **sorted** within their level in descending order using a *severity* metric. In order to produce the metric value, an *importance* function is required.

Let $imp_A : A \to \{q \in \mathbb{Q} \mid 0 \leq q \leq 1\}$ be a function, that returns *importance* of the given activity. Let $imp_V : V_L \to \{q \in \mathbb{Q} \mid 0 \leq q \leq 1\}$ be a function, that returns the *importance* of the given attribute. This importance is an empirical value between 0 and 1, which allows to influence position of specific problems in the list (the higher the value, the higher the position). It is assumed to be the part of the input.

The general formula for calculation *severity* of any item is $severity = factor * cardinality$, where *factor* is a specific factor, calculated in the way described in Table 4.4 and *cardinality* is the cardinality of the multiset, corresponding to the item. In other words, *severity* of an item, is the cardinality of the corresponding set, weighted by the importance of the activity and the attribute defining the set.

Items $pli_{(v,u)}$ have slightly more complicated way of calculating the *factor*. In order to calculate it, we need to define an additional function. Given an activity name $aname \in A$, an attribute $v \in V_L^{context}$, and its value $u \in G_v$, let $occur : A \times V_L^{context} \times U_L \rightarrow N$ be a function, assigning number of occurrences of steps with given activity name in the set $S_{assignment}(v, u)$, i.e. $occur(aname, v, u) =\mid \{z \in S_{assignment}(v, u) \mid sname(z) = aname\} \mid$.

Table 4.4: Severity metric

Item	Factor	Corresponding multiset
pli_{aname}	$imp_A(aname)$	$S_{anyviolation}(aname)$
$pli^{aname}_{model.only}$	$imp_A(aname)$	$S_{model.only}(aname)$
$pli^{aname,(v,u)}_{model.only}$	$imp_A(aname) * imp_V(v)$	$S_{model.only.assignment}(aname, v, u)$
$pli^{aname}_{log.only}$	$imp_A(aname)$	$S_{log.only}(aname)$
$pli^{aname,(v,u)}_{log.only}$	$imp_A(aname) * imp_V(v)$	$S_{log.only.assignment}(aname, v, u)$
$pli^{aname}_{wrong.w}$	$imp_A(aname)$	$S_{dataviolation}(aname, w)$
$pli^{aname,(v,u)}_{wrong.w}$	$imp_A(aname) * imp_V(v)$	$S_{dataviolation.assignment}(aname, v, u)$
$pli_{(v,u)}$	$imp_V(v) * \sum_{aname \in A}(occur(aname, v, u) * imp_A(aname))$	$S_{assignment}(v, u)$
$pli^{(v,u),aname}_{model.only}$	$imp_A(aname) * imp_V(v)$	$S_{assignment.model.only}(aname, v, u)$
$pli^{(v,u),aname}_{log.only}$	$imp_A(aname) * imp_V(v)$	$S_{assignment.log.only}(aname, v, u)$
$pli^{(v,u),aname}_{wrong.w}$	$imp_A(aname) * imp_V(v)$	$S_{assignment.dataviolation}(aname, v, u, w)$

The resulting list is sorted using the *severity* values of each item in descending order. Please note that the items are sorted independently within each tuple.

4.3.3 Refining the List

The obtained list might appear to have a considerable size and become incomprehensible. In order to overcome this, the list is **refined**. First, all items corresponding to the empty sets are removed. Second, all items, which have $CPIR_{CF} < CPIR^{min}_{CF}$ and $CPIR_{CF} > CPIR^{max}_{CF}$, i.e. items whose relevance is not within the specified bounds, are removed. As was mentioned, the bounds $CPIR^{min}_{CF}$ and $CPIR^{max}_{CF}$ are specified by the user.

In this section we describe how relevance metric is calculated. The metric is named $CPIR_{CF}$ and is based on the *CPIR* metric described in Section 2.2. We are also going to define $supp_{CF}$ as an adaptation of *supp* for our approach. The metrics allow us to quantify how context attribute values, like purchased amount name or executor are connected to the observed violations. The closer the absolute value of $CPIR_{CF}$ to 1, the more it is likely that the context is connected to the violation and vice versa the closer the value to 0, the more likely it is that there is no connection between the violation and the attribute assignment.

In order to define the metrics, we assume the given multiset of *Data-Aware Steps* Z, the itemset I based on Z and the multiset of transactions D based on the steps in Z.

We define $supp_{CF}(K)$, $K \subseteq Z$ as $supp_{CF}(K) = \frac{|K|}{|Z|}$. For example, $supp_{CF}(R_{violation}) = \frac{|R_{violation}|}{|Z|}$.

We define $CPIR_{CF}$ individually for all types of items and sub-items of our list. Given an activity name $aname \in A$ we compute strength of the relation between presence of the activity name and:

- any kind of violation:

$CPIR_{CF}(pli_{aname}) =$

$$\frac{supp_{CF}(S_{anyviolation}(aname)) - (supp_{CF}(R_{name}(aname)) * supp_{CF}(R_{violation}))}{supp_{CF}(R_{name}(aname)) * (1 - supp_{CF}(R_{violation}))}$$

- move on model:
  $CPIR_{CF}(pli^{aname}_{model.only}) =$

$$\frac{supp_{CF}(S_{model.only}(aname)) - (supp_{CF}(R_{name}(aname)) * supp_{CF}(R_{model.only}(aname)))}{supp_{CF}(R_{name}(aname)) * (1 - supp_{CF}(R_{model.only}(aname)))}$$

- move on log:
  $CPIR_{CF}(pli^{aname}_{log.only}) =$

$$\frac{supp_{CF}(S_{log.only}(aname)) - (supp_{CF}(R_{name}(aname)) * supp_{CF}(R_{log.only}(aname)))}{supp_{CF}(R_{name}(aname)) * (1 - supp_{CF}(R_{log.only}(aname)))}$$

Given an activity name *aname* $\in A$ and a specification attribute $w \in V_M$, we compute the strength of the relation between presence of the activity name and move with incorrect w:
$CPIR_{CF}(pli^{aname}_{wrong.w}) =$

$$\frac{supp_{CF}(S_{dataviolation}(aname, w)) - (supp_{CF}(R_{name}(aname)) * supp_{CF}(R_{log.only}(aname)))}{supp_{CF}(R_{name}(aname)) * (1 - supp_{CF}(R_{incorrect.attr}(w)))}$$

Given an attribute $v \in V_L^{context}$ and its value $u \in G_v$, we compute the strength of the relation between the attribute assignment and violations of any type:
$CPIR_{CF}(pli_{(v,u)}) =$

$$\frac{supp_{CF}(S_{assignment}(v,u)) - (supp_{CF}(R_{val}(v,u)) * supp_{CF}(R_{violation}))}{supp_{CF}(R_{val}(v,u)) * (1 - supp_{CF}(R_{violation}))}$$

Given an activity name *aname* $\in A$, an attribute $v \in V_L^{context}$, its value $u \in G_v$ we compute strength the following (the higher the value, the more likely it is that the presence of the violation implies the attribute value):

- Strength of the relation between the presence of a **move on model** for the activity *aname* and the attribute assignment (v, u):
  $CPIR_{CF}(pli^{aname,(v,u)}_{model.only}) =$

$$\frac{supp_{CF}(S_{model.only.assignment}(aname, v, u)) - (supp_{CF}(R_{val}(v,u)) * supp_{CF}(R_{model.only}(aname)))}{supp_{CF}(R_{model.only}(aname)) * (1 - supp_{CF}(R_{val}(v,u)))}$$

- Strength of the relation between the presence of a **move on log** for the activity *aname* and the attribute assignment (v, u):
  $CPIR_{CF}(pli^{aname,(v,u)}_{log.only}) =$

$$\frac{supp_{CF}(S_{log.only.assignment}(aname, v, u)) - (supp_{CF}(R_{val}(v,u)) * supp_{CF}(R_{log.only}(aname)))}{supp_{CF}(R_{log.only}(aname)) * (1 - supp_{CF}(R_{val}(v,u)))}$$

- Strength of the relation between the presence of a **move on with incorrect** w (given a specification attribute $w \in V_M$) for the activity *aname* and the attribute assignment (v, u):
  $CPIR_{CF}(pli^{aname,(v,u)}_{wrong.w}) =$

$$\frac{supp_{CF}(S_{dataviolation.assignment}(aname, v, u, w)) - (supp_{CF}(R_{val}(v,u)) * supp_{CF}(R_{incorrect.attr}(aname, w)))}{supp_{CF}(R_{incorrect.attr}(aname, w)) * (1 - supp_{CF}(R_{val}(v,u)))}$$

Given the same input, we compute strength of the opposite relations (the higher the value, the more likely it is that the attribute value implies the presence of the violation):

- Strength of the relation between the presence of the attribute assignment (v, u) and a **move on model** for the activity *aname*:
  $CPIR_{CF}(pli^{(v,u),aname}_{model.only}) =$

$$\frac{supp_{CF}(S_{assignment.model.onlY}(aname,v,u))-(supp_{CF}(R_{model.only}(aname))*supp_{CF}(R_{val}(v,u)))}{supp_{CF}(R_{val}(v,u))*(1-supp_{CF}(R_{model.only}(aname)))}$$

- Strength of the relation between the presence of the attribute assignment (v, u) and a **move on log** for the activity *aname*:
  $CPIR_{CF}(pli^{(v,u),aname}_{log.only}) =$

$$\frac{supp_{CF}(S_{assignment.log.only}(aname,v,u))-(supp_{CF}(R_{log.only}(aname))*supp_{CF}(R_{val}(v,u)))}{supp_{CF}(R_{val}(v,u))*(1-supp_{CF}(R_{log.only}(aname)))}$$

- Strength of the relation between the presence of the attribute assignment (v, u) and a **move on with incorrect** w (given a specification attribute $w \in V_M$) for the activity *aname*:
  $CPIR_{CF}(pli^{(v,u),aname}_{wrong.w}) =$

$$\frac{supp_{CF}(S_{assignment.dataviolation}(aname,v,u,w))-(supp_{CF}(R_{incorrect.attr}(aname,w))*supp_{CF}(R_{val}(v,u)))}{supp_{CF}(R_{val}(v,u))*(1-supp_{CF}(R_{incorrect.attr}(aname,w)))}$$

To demonstrate the described metrics ($CPIR_{CF}$ and *severity*), we show their values for the previously introduced example. The *severity* values are computed assuming importance of every activity and attribute is 0.5. Please note that the $CPIR_{CF}$ values are not reliable since the dataset is extremely small. Yet, the example is able to illustrate the way the values are obtained.

- When AR = *5000* ($CPIR_{CF} = 0.25$; *severity* = 0.75);
  - When AR = *5000*, activity c is performed with incorrect AR ($CPIR_{CF} = 0.077$; *severity* = 0.25);
  - When AR = *5000*, activity e is performed with incorrect *Ex* ($CPIR_{CF} = 0.077$; *severity* = 0.25);
  - When AR = *5000*, activity h is performed with incorrect *Time* ($CPIR_{CF} = 0.077$ *severity* = 0.25);
- c ($CPIR_{CF} \approx 0.4375$; *severity* = 0.5);
  - Activity c is performed with incorrect AR ($CPIR_{CF} \approx 0.4375$; *severity* : 0.5);
    - Activity c is performed with incorrect AR, when *Ex* = *Mike* ($CPIR_{CF} = 1.0$; *severity* : 0.25);
    - Activity c is performed with incorrect AR, when AR = *5000* ($CPIR_{CF} = 1.0$; *severity* : 0.25);
- e ($CPIR_{CF} \approx 0.25$; *severity* = 0.5);
  - Activity e is performed with incorrect *Ex* ($CPIR_{CF} \approx 0.25$; *severity* = 0.5);
    - Activity e is performed with incorrect *Ex*, when *Ex* = *Mike* ($CPIR_{CF} = 1.0$; *severity* : 0.25);
    - Activity e is performed with incorrect *Ex*, when AR = *5000* ($CPIR_{CF} = 1.0$; *severity* : 0.25);
- h ($CPIR_{CF} \approx 0.4375$; *severity* = 0.5);
  - Activity h is performed with incorrect *Time* ($CPIR_{CF} \approx 0.4375$; *severity* = 0.5);
    - Activity h is performed with incorrect *Time*, when *Ex* = *Pete* ($CPIR_{CF} \approx 1.0$; *severity* = 0.25);

        * Activity h is performed with incorrect *Time*, when $AR = 5000$ ($CPIR_{CF} \approx 1.0$; *severity* : 0.25);

- When $Ex = Mike$ ($CPIR_{CF} = 1.0$; $severity = 0.5$);
    - When $Ex = Mike$, activity c is performed with incorrect AR ($CPIR_{CF} \approx 0.48$; $severity = 0.25$);
    - When $Ex = Mike$, activity e is performed with incorrect Ex ($CPIR_{CF} \approx 0.48$; $severity = 0.25$);
- When $Ex = Pete$ ($CPIR_{CF} \approx 0.069$; $severity = 0.25$);
    - When $Ex = Pete$, activity h is performed with incorrect *Time* ($CPIR_{CF} \approx 0.0135$; $severity = 0.25$);

Chapter 5

Investigating specific problems

In Chapter, 4 we presented our approach for obtaining violation statistics and a list of problems. In this chapter we illustrate how we obtain detailed insight into a single problem from a previously obtained list. This step (Figure 5.1) offers a deeper analysis of the root cause of a particular problem. This is achieved by analyzing all aspects of the Data-Aware Steps where the problem occurs and comparing them to those where the problem does not occur. The inputs for this approach phase are the Data-Aware Replay Result, the list of considered attributes and the problem description from the list obtained from the previous step. The output is a list of the root causes of the analyzed problem.

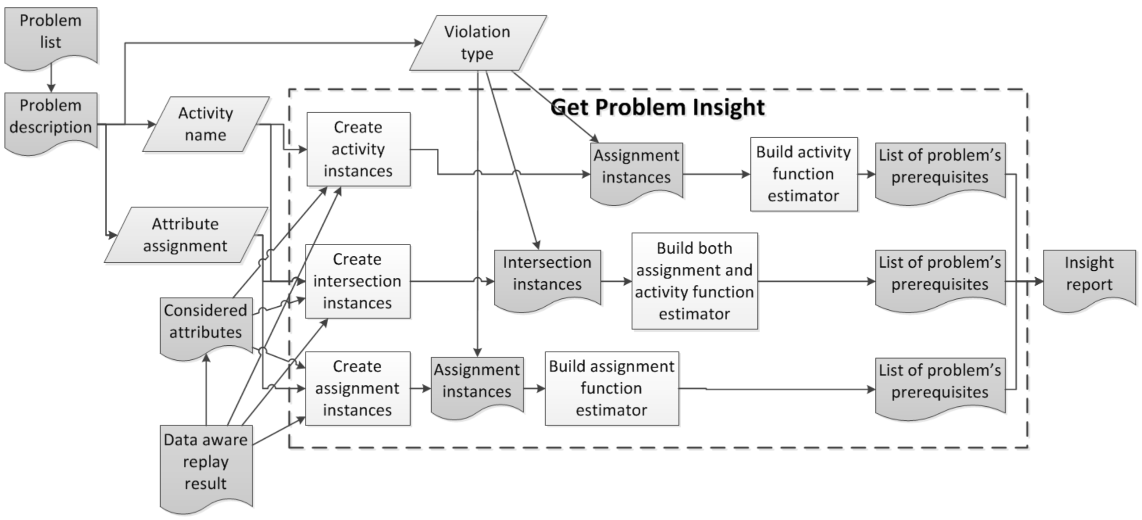

Problem list
Problem description
Violation type
Activity name
Attribute assignment
Considered attributes
Data aware replay result
Get Problem Insight
Create activity instances
Create intersection instances
Create assignment instances
Assignment instances
Intersection instances
Assignment instances
Build activity function estimator
Build both assignment and activity function estimator
Build assignment function estimator
List of problem's prerequisites
List of problem's prerequisites
List of problem's prerequisites
Insight report

Figure 5.1: Approach step 2

The input consists of several parts. First, a specific item from the problem list is provided as an input. Thus, the specific violation type is always present in the description.

Another input is a set of attributes that we want to include in second phase analysis (root-cause investigation). It can only be a subset of those present in the analyzed event log.

Finally, the Data-Aware Replay Result contains Data-Aware Steps based on which three different instance sets from the Data-Aware Steps are produced. There are up to 3 instance sets created (based on the information present in problem description):

- instances corresponding to Data-Aware Steps with given activity name $aname \in A$;
- instances corresponding to Data-Aware Steps with given attribute assignment $v \in V_L^{context}$, $u \in G_v$;

- instances corresponding to Data-Aware Steps with both given activity name $aname \in A$ and attribute assignment $v \in V_L^{context}$, $u \in G_v$.

If either an activity name or an attribute assignment is not present in the problem description, only one set of instances is built. Each set of instances is used to build a separate function estimator and obtain a list of problem prerequisites.

Further in this chapter we explain the approach phase in more detail. Section 5.1 gives detailed information about the input. Section 5.2 explains how instance sets and function estimators are obtained and Section 5.3 explains what is produced as a result.

5.1 Input

In this section, we describe the input for the approach phase in more detail.

The first input is a list of problems, described in Section 4.3.1. From this list a single item is selected for the analysis. List items of the first level cannot be an input for the approach phase, since they do not refer to a specific problem. Instead, they refer to all problems, that occur with the corresponding activity name or attribute assignment. Basically, each list item and sub-item in the list has up to 3 variable parts: an *activity name* $aname \in A$, an *attribute assignment* $(v, u) \in (V_L^{context} \times G_v)$ and a *violation type*, which is either *move on log*, *move on model*, or *move with incorrect* w, where $w \in V_M$. The problem description corresponding to the item is used for identifying the subset of data aware replay steps to analyze. Since input item **must not** belong the first level of the list, violation type and activity name are always present in the problem description, while attribute assignment is optionally present.

For example, let us consider the item $pli_{wrong.AR}^{(Ex,Mike),c}$, which is described as: When $Ex = Mike$, activity c is performed with incorrect AR. This description has the following variable parts: the activity name c, the attribute assignment $(Ex, Mike)$ and the violation type *move with incorrect* AR.

The second input is a Data-Aware Replay Result DR in the form described in Section 2.1.3 as well as a set Z, containing all the Data-Aware Steps from DR. For the demonstration, we are going to use the Data-Aware Replay Result, shown in Table 3.4.

5.2 Comparing Steps Within Sets

Based on activity name and attribute value, up to 3 subsets of the Data-Aware Steps are identified. The examples assume that the selected item is $pli_{wrong.AR}^{(Ex,Mike),c}$, described as 'When $Ex = Mike$, activity c is performed with incorrect AR'.

1. steps with the given activity name, i.e. $\{z \in Z \mid sname(z) = aname\}$, if $aname$ is defined for the problem. For our example, $aname = c$ and the subset will be: $\{z_{15}, z_{24}\}$;

2. steps with the given attribute assignment, i.e. $\{z \in Z \mid val(z, v) = u\}$, if v and u are defined for the problem. For our example, $v = Ex$ and $u = Mike$ and the subset will be: $\{z_{24}, z_{26}\}$;

3. steps with both the given activity name and attribute assignment, i.e. $\{z \in Z \mid sname(z) = aname \wedge val(z, v) = u\}$ if $aname$, v and u are defined for the problem. For our example the subset will be: $\{z_{24}\}$.

These subsets will serve as a basis for creation sets of observation instances, described further.

Discovery of the root causes of the problem can be translated into the problem of finding the best estimator of a function.

Let $f : A_1 \times A_2 \times \ldots \times A_n \to B$ be a function having a finite domain B. Similarly to [18], we define **estimator of function** f as a function $\psi_f : B \to 2^{A_1 \times A_2 \times \ldots \times A_n}$, such that, for each $b \in B$, $\psi_f(b)$ returns all input domain tuples for which the expected output is b.

Function estimator can be found using a decision tree algorithm. There is a number of decision tree algorithms available. The most popular include: CHAID, ID3, CART, MARS, C4.5 [1]. The decision tree building algorithm C4.5 [23] is used to build the function estimator in our approach. Choosing this algorithm is motivated by the fact that the input is likely to contain errors (e.g. wrong timestamps) and missing values (e.g. if log does not contain information about ID of a user, responsible for some events). Besides, some of the input attribute domains are very likely to be continuous. The brief description of the algorithm is provided in Appendix A.

The decision tree will be trained through a set of observations. An *observation instance* is a pair $oi = (\vec{a}, b)$, where $\vec{a} \in A_1 \times A_2 \times \ldots \times A_n$ is the *observed input*, b is the *observed output*. The instance is based on a single step in an alignment.

In order to derive the observation instance from the Data-Aware Step, a set of *considered features* $V_{considered}$ should be selected. The set determines what values comprise the observed input of the instance. $V_{considered} \subseteq V_L^{context} \cup \{activity\_name\} \cup TF$, where *activity_name* is an indication, whether the activity name of the step is considered. $TF = \{tIgnore, tFirst, tPrevious, t_{aname_1}, \ldots, t_{aname_n}\}$, where $aname_i \in A$. Exactly one member of TF must be included into $V_{considered}$, which determines the way the *time feature* is considered.

In order to define the *time feature*, we introduce the notion of *instance step*. The *instance step* is a Data-Aware Step z from which the instance oi_z is derived. Let t_z be the time of occurrence of z (i.e. t_z is the timestamp value of z). Let t_a be the time of occurrence of *anchor step*. Then, the time feature value for the instance oi_z is $t_f = t_z - t_a$. The *anchor step* is determined by the element of TF, which is included into $V_{considered}$.

If $tIgnore \in V_{considered}$, the *time feature* is ignored. Otherwise, the *anchor step* is one of the following:

- the first step in the alignment, if $tFirst \in V_{considered}$;
- the step, previous to the instance step, if $tPrevious \in V_{considered}$;
- the latest occurrence of the step with the specified activity name *aname*, that happened before the *instance step*, if $t_{aname} \in V_{considered}$.

Assuming $V_{considered} = V_L^{context} \cup \{activity\_name\} \cup \{tFirst\}$, $\vec{a} \in A \times G_{v_1} \times G_{v_2} \times \ldots \times G_{v_n} \times \mathbb{Q}_0^{+}$[1], where $v_i \in V_L^{context}, i = 1, 2, \ldots, |\ V_L^{context}\ |$.

Our observed output b is an indicator, whether the step, corresponding to the instance is a violation of the type, described in the input problem description. Thus, $b \in \{problematic, nonproblematic\}$.

We show the example, assuming the Data-Aware Replay Result shown in Table 3.4, the selected item $pli_{wrong.AR}^{(Ex,Mike),c}$ from the example list shown in Section 4.3.3, and the set of considered features $V_{considered} = \{activity\_name, AR, Ex, tPrevious\}$

1. Instances, corresponding to the steps with the given activity name:
   $\{(\boldsymbol{c}, 3000, Ellen, 1800, nonproblematic)(\boldsymbol{c}, 5000, Mike, 1000, problematic)\}$
2. Instances, corresponding to the steps with the given attribute assignment:
   $\{(c, 5000, \boldsymbol{Mike}, 1000, problematic), (e, 5000, \boldsymbol{Mike}, 200, nonproblematic)\}$
3. Instances, corresponding to the steps with the given activity name and attribute assignment:
   $\{(\boldsymbol{c}, 5000, \boldsymbol{Mike}, 1000, problematic)\}$

5.3 Output

As described in [18], decision trees classify instances by sorting them down in a tree from the root to some leaf node. Each non-leaf node specifies a test of some attribute $v_1, \ldots, v_n$ and each branch descending from that node corresponds to a range of possible values for this attribute. In general, a decision tree represents a disjunction of conjunctions of expressions: each path from the tree root

[1] $\mathbb{Q}_0^+ = \{q \in \mathbb{Q} \mid q \geq 0\}$.

to a leaf corresponds to an expression that is, in fact, a conjunction of attribute tests. Each leaf node is associated with one of the possible output values: if an expression *e* is associated with a path to a leaf node *b*, every input tuple for which e evaluates to true is expected to return y as output.

The output of this step is a decision tree, with leaves corresponding either to the subset with selected problem, or without it.

If we use the example from the previous section, we would obtain the following trees:

1. The tree, based on instances, corresponding to the steps with the given activity name
   - *Ex = Mike*
     - *problematic*
   - *Ex = Ellen*
     - *nonroblematic*
2. The tree, based on instances, corresponding to the steps with the given attribute assignment
   - *Activity name = c*
     - *problematic*
   - *Activity name = e*
     - *nonproblematic*
3. The tree, based on instances, corresponding to the steps with both the given activity name and the given attribute assignment. is not very informative, since the example set of instances consists of a single instance and there are no other instances to compare with.

Chapter 6

Design and implementation

In chapters 3, 4 and 5 we described how we detect violations, produce a list of problems with diagnostics and get a detailed insight into selected problems from the list. In this chapter, we explain the architecture of our analysis software and the design decisions made. This chapter also covers the question raised in Chapter 1: *visualizing the resulting diagnostics in a way understandable for users without a technical background.*

The approach is implemented as a number of plugins for the process mining tool ProM[1]. Figure 6.1 shows correspondence between the parts of the approach and the implemented plugins.

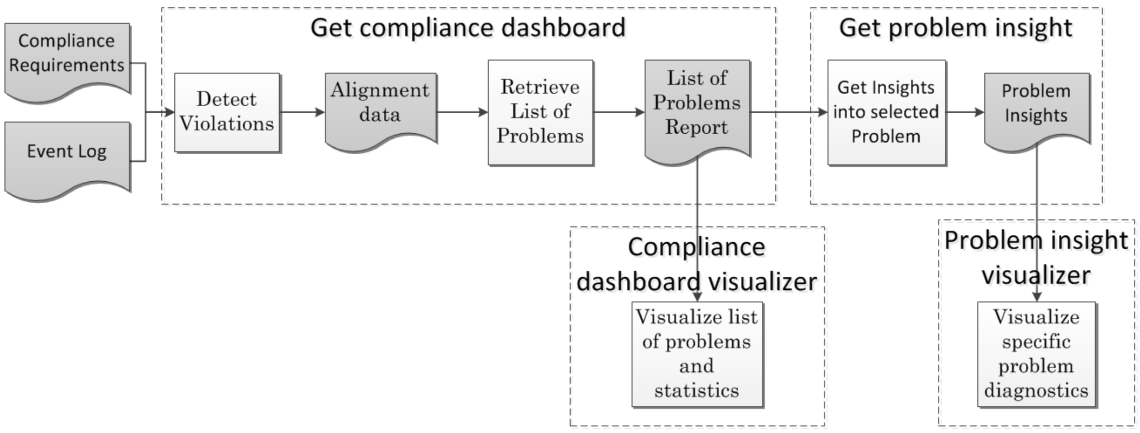

Compliance Requirements
Event Log
Get compliance dashboard
Detect Violations
Alignment data
Retrieve List of Problems
List of Problems Report
Get problem insight
Get Insights into selected Problem
Problem Insights
Compliance dashboard visualizer
Visualize list of problems and statistics
Problem insight visualizer
Visualize specific problem diagnostics

Figure 6.1: Implementation of the parts of the approach

As is seen in the figure, the input Petri Net with Data is implemented within the DataPetriNets package, while the event log is provided by the OpenXES library. Obtaining alignments and retrieving list of problems is implemented within the 'Get compliance dashboard' plugin, while obtaining insight of a specific problem is implemented within the 'Get problem insight' plugin. Outputs of the two mentioned plugins are visualized by the 'Compliance dashboard visualizer' and the 'Problem insight visualizer' plugins respectively. The remainder of this chapter is organized as follows. The 'Get compliance dashboard' plugin will be introduced in Section 6.1. Section 6.2 introduces the 'Get problem insight'.

[1]ProM is an extensible framework that supports a wide variety of process mining techniques in the form of plugins (`http://www.promtools.org/prom6/`)

6.1 Compliance Dashboard Plugin

The plugin implements the first two steps of the approach, described in chapters 3 and 4. In general, inputs and outputs are shown in the Table 6.1.

Table 6.1: ‘Get compliance dashboard’ inputs and outputs

Feature	**Description**
Plug-in Name	Get Compliance dashboard
Input	(1) Event log (2) Compliance rule, represented as a Petri Net with Data
Output	Compliance dashboard with violation statistics and report

An event log should be represented in XES format. The rules are specified in terms of a Petri Net with Data. One of the possible ways to define a Petri Net with Data so that it can be used in ProM is to follow the listed steps:

1. First, Control-Flow rules should be defined in the form of a Petri Net. It can be represented in the form of a PNML[2] file, which is supported, among other tools, by YASPER[3].
2. After that, data and resource rules should be added to the Petri Net, which would result in a Petri Net with Data. This can be done using 'Create/edit PetriNet with Data' plugin in ProM.
3. The temporal constraints are defined separately using the ‘Get compliance dashboard’ plugin.

6.1.1 Integration into the Existing ProM Environment

The implementation of the plugin involves interaction between multiple modules. These are shown in Figure 6.2, as well as steps of the approach implemented within each module.

Since control flow compliance checking techniques are already implemented within the 'Check compliance using conformance checking' plugin mentioned in [25], it is reused in our approach for the corresponding part.

The Data-Aware Conformance Checker was implemented as a part of this thesis. It is responsible for identifying the set of admissible values for each activity based on the assigned guard.

As it is shown, the data-aware conformance checker is responsible for checking compliance of the data and resource perspectives. It can also be reused for temporal compliance checking, with the difference that the log should be enriched first.

Finally, after obtaining a Data-Aware Replay Result, the Compliance Dashboard component does the rest of the job as was described in Chapter 4.

6.1.2 Graphical User Interface

Running the Plugin

After running the plugin, user has to define a mapping between the log and the model (Figure 6.3). If an event class is not mapped to any activity in the model, the log will be abstracted from it, i.e. such events will be mapped to an activity named Ω.

If the user defined attributes in the DP-net, they would be offered to setup the variable mapping the same way as for Conformance checking of Petri Net with Data(Figure 6.4).

[2]http://www.pnml.org/

[3]http://www.yasper.org/

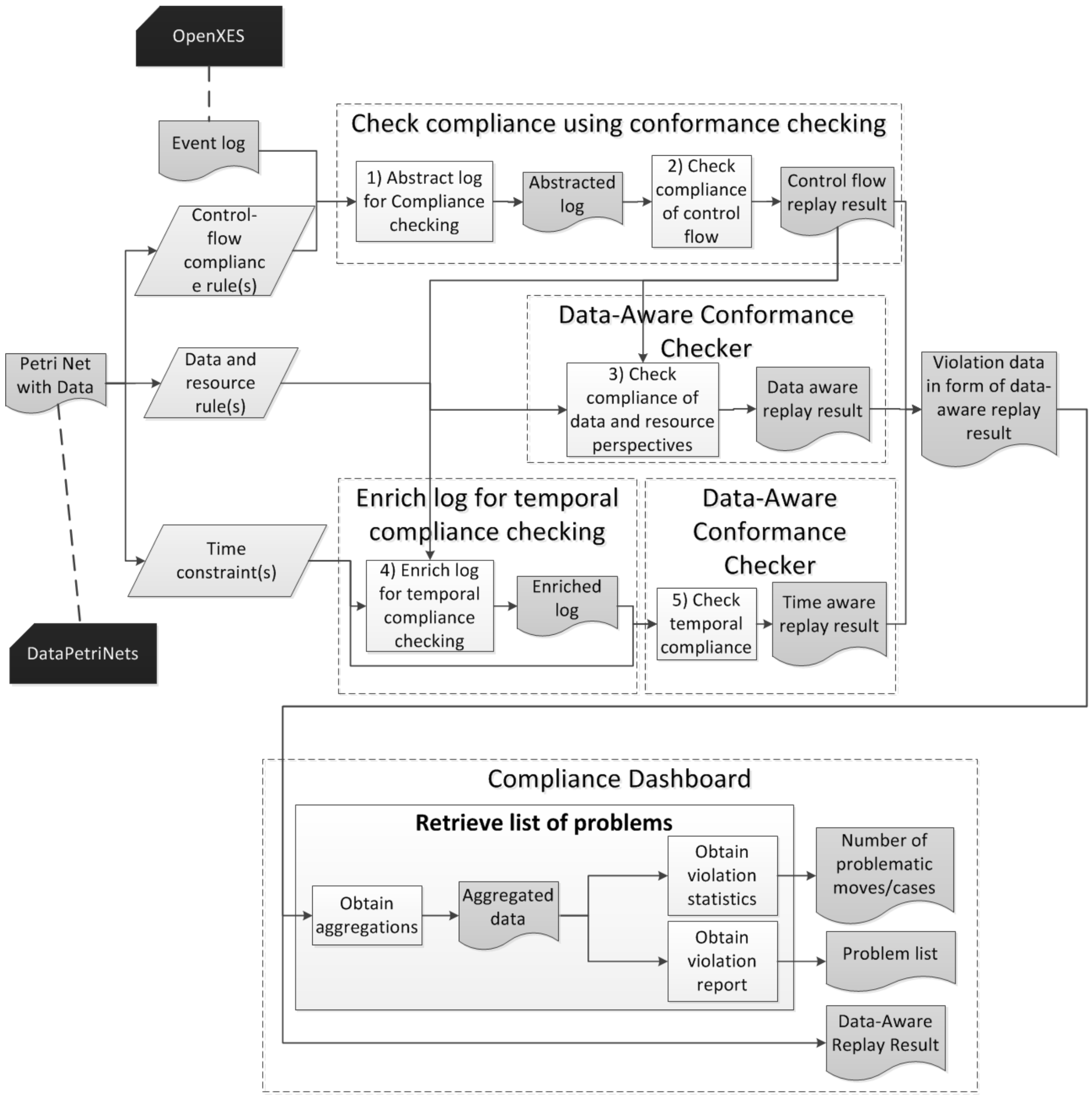

OpenXES
Event log
Check compliance using conformance checking
1) Abstract log for Compliance checking
Abstracted log
2) Check compliance of control flow
Control flow replay result
Control-flow complianc e rule(s)
Petri Net with Data
Data and resource rule(s)
Data-Aware Conformance Checker
3) Check compliance of data and resource perspectives
Data aware replay result
Violation data in form of data-aware replay result
Time constraint(s)
Enrich log for temporal compliance checking
4) Enrich log for temporal compliance checking
Enriched log
Data-Aware Conformance Checker
5) Check temporal compliance
Time aware replay result
DataPetriNets
Compliance Dashboard
Retrieve list of problems
Obtain aggregations
Aggregated data
Obtain violation statistics
Obtain violation report
Number of problematic moves/cases
Problem list
Data-Aware Replay Result

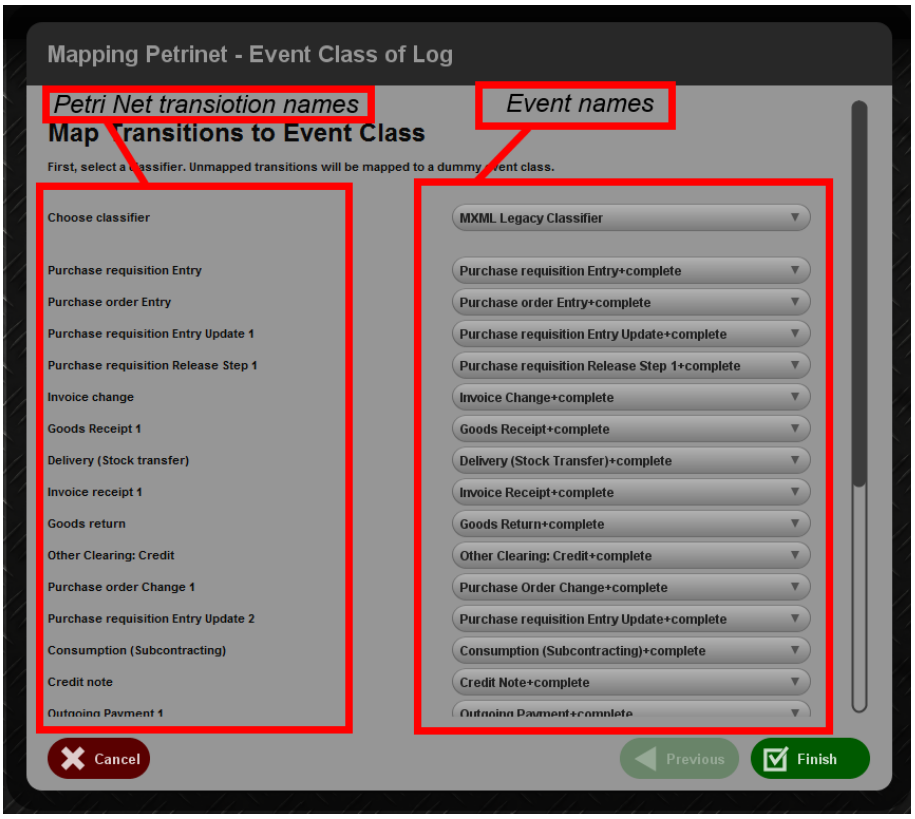
Figure 6.2: Interactions within the 'Get compliance dashboard' plugin

Then the user is offered the option to perform temporal compliance checking. If this option is taken, the temporal pattern needs to be defined, which includes defining guards the same way as for data rules, with the exception that the rules are defined for timestamps instead (Figure 6.5). When the plugin completes, user can see the visualization.

Plugin Output Visualization

This visualization consists of 2 tabs (Figure 6.6). In the 'Move statistics' tab the user can assess different figures:

- total number of moves,
- number of invisible moves,
- number of moves on log,


Mapping Petrinet - Event Class of Log
Petri Net transiotion names
Event names
Cancel
Previous
Finish

Map ransitions to Event Class

First, select a assifier. Unmapped transitions will be mapped to a dummy vent class.

Choose classifier	MXML Legacy Classifier
Purchase requisition Entry	Purchase requisition Entry+complete
Purchase order Entry	Purchase order Entry+complete
Purchase requisition Entry Update 1	Purchase requisition Entry Update+complete
Purchase requisition Release Step 1	Purchase requisition Release Step 1+complete
Invoice change	Invoice Change+complete
Goods Receipt 1	Goods Receipt+complete
Delivery (Stock transfer)	Delivery (Stock Transfer)+complete
Invoice receipt 1	Invoice Receipt+complete
Goods return	Goods Return+complete
Other Clearing: Credit	Other Clearing: Credit+complete
Purchase order Change 1	Purchase Order Change+complete
Purchase requisition Entry Update 2	Purchase requisition Entry Update+complete
Consumption (Subcontracting)	Consumption (Subcontracting)+complete
Credit note	Credit Note+complete
Outgoing Payment 1	Outgoing Payment+complete

Figure 6.3: Log-model mapping

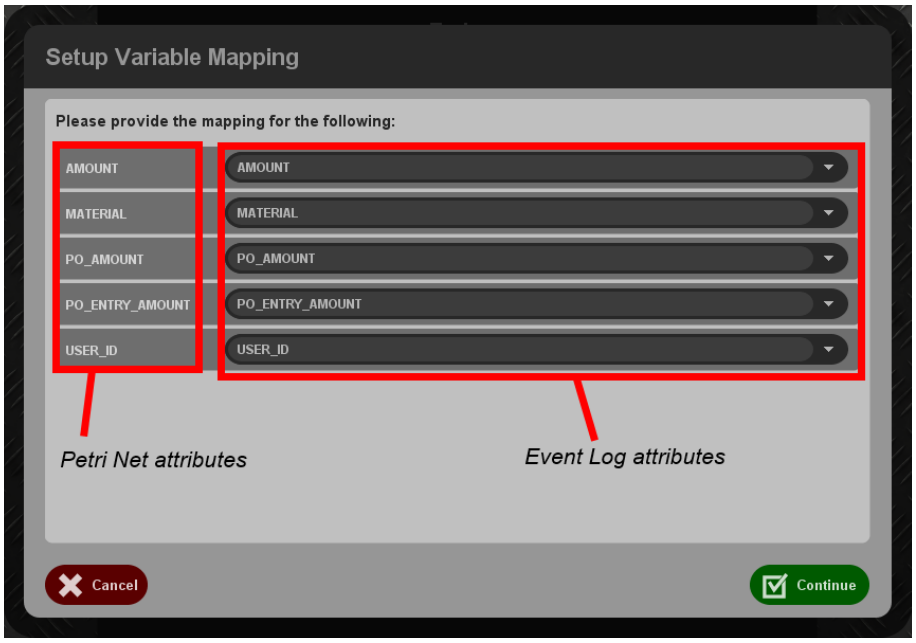
- number of moves on model,
- number of synchronous moves (i.e. where neither activity name, nor event name is $\gg$),
- number of synchronous moves with correct attribute for each attribute,
- number of synchronous moves with all attributes correct,
- number of synchronous moves with incorrect attribute for each attribute, and
- number of synchronous moves with all attributes incorrect.

The statistics are represented in the form of a table, where each row corresponds to a single activity, or a value of an attribute (or an interval for continuous attributes), depending on the user choice (Figure 6.6). The choice is made using a combobox at the top of the left panel(1). The panel also allows the user to hide/show table columns. The columns in the table (2) are color coded: those corresponding to problematic moves/cases are red while those corresponding to non-problematic moves/cases are green. When the user clicks on a table row, the bar chart (3) visualizes values in that column.

By clicking on the header of a column it is possible to sort the table by the values in the column. Repeating the click sorts in the reverse order.

It is possible to hide columns that are not of interest by unchecking a checkbox with the corresponding column name. The values in the rows are defined by the combobox, whose value can be either an activity or one of the attributes defined in the Petri Net.

The second tab contains a problem report (Figure 6.7). The report consists of items of the form described in Section 4.3.1 (1). The items corresponding to activities are marked with the ▶


Setup Variable Mapping

Please provide the mapping for the following:

AMOUNT	AMOUNT
MATERIAL	MATERIAL
PO_AMOUNT	PO_AMOUNT
PO_ENTRY_AMOUNT	PO_ENTRY_AMOUNT
USER_ID	USER_ID

Figure 6.4: Mapping between log attributes and Petri Net attributes

icon, while those corresponding to attribute assignments are marked with the ▤ icon. Each item can be expanded, enabling the user to learn more context details regarding the problem.

The interface has a number of controls that allow the user to sort and refine items in the report (2). First, the user can filter out items with assignments of all attributes, except the one selected using 'attribute filter'. Second, the user can change ranking of the items by changing the importance of activities and attributes. Finally, the interface allows the user to remove all items, whose CPIR value does not lie within the defined minimal and maximal boundaries.

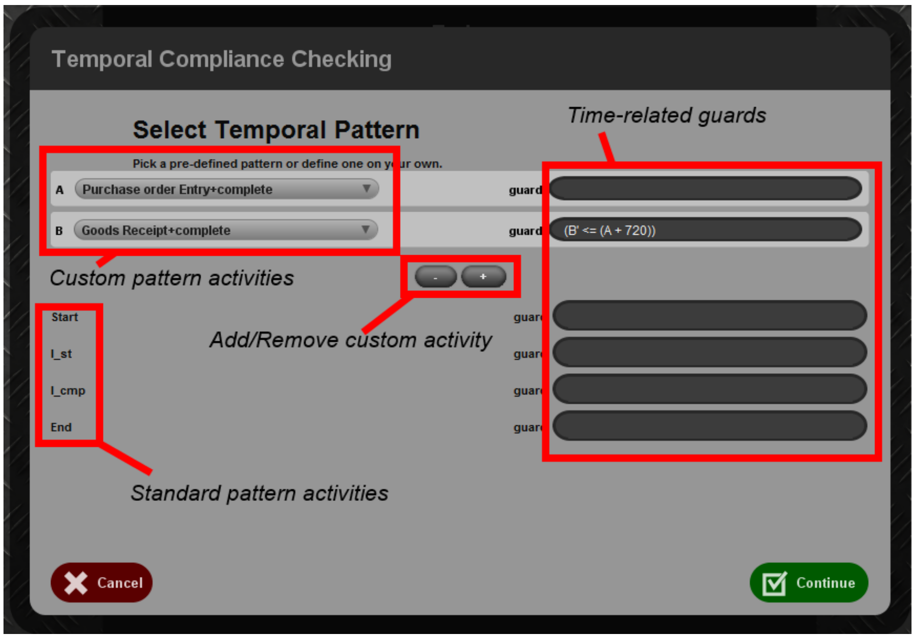
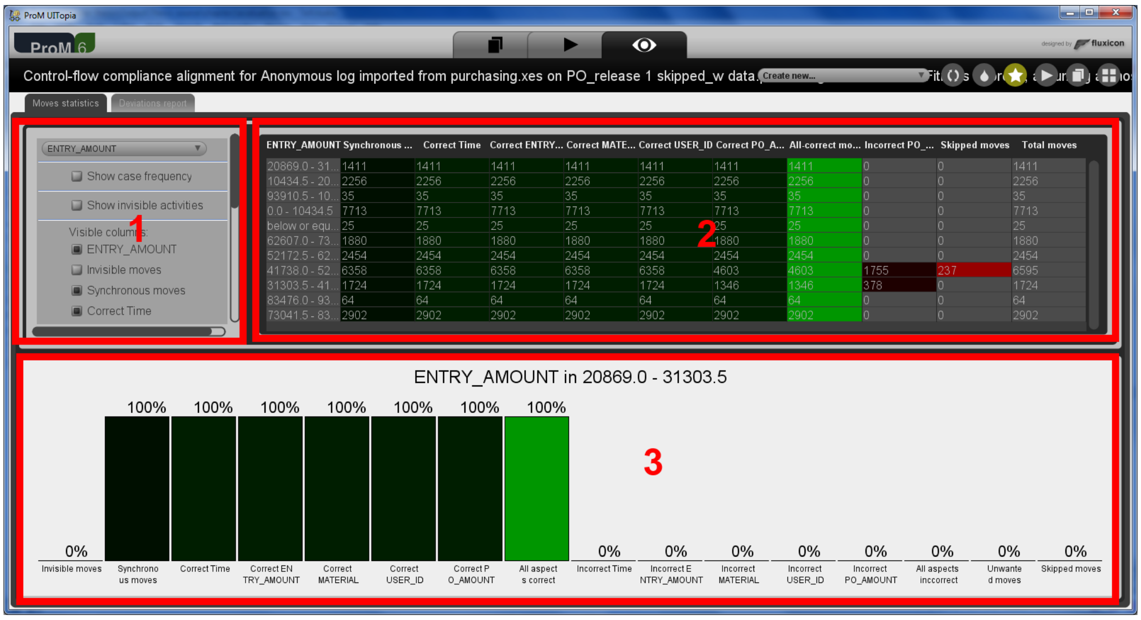
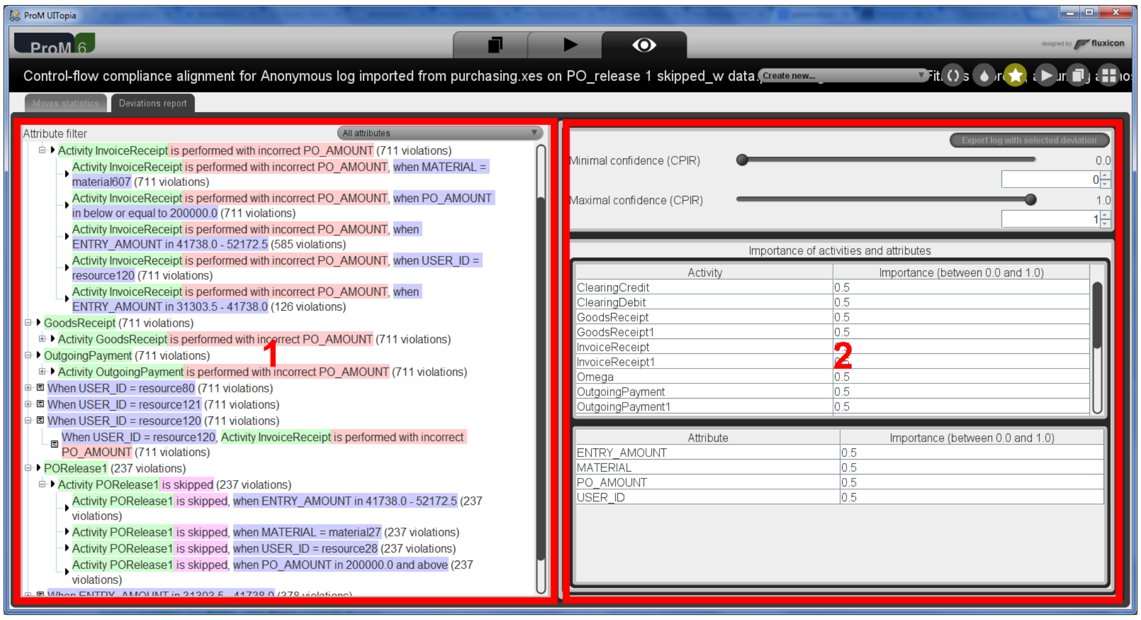

Temporal Compliance Checking
Select Temporal Pattern
Time-related guards
Pick a pre-defined pattern or define one on your own.
A Purchase order Entry+complete
B Goods Receipt+complete
guard
guard (B' <= (A + 720))
Custom pattern activities
- +
Start
I_st
I_cmp
End
Add/Remove custom activity
Standard pattern activities
Cancel
Continue

Figure 6.5: Defining of the temporal pattern



ENTRY_AMOUNT	Synchronous ...	Correct Time	Correct ENTRY...	Correct MATE...	Correct USER_ID	Correct PO_A...	All-correct mo...	Incorrect PO_...	Skipped moves	Total moves
20869.0 - 31...	1411	1411	1411	1411	1411	1411	1411	0	0	1411
10434.5 - 20...	2256	2256	2256	2256	2256	2256	2256	0	0	2256
93910.5 - 10...	35	35	35	35	35	35	35	0	0	35
0.0 - 10434.5	7713	7713	7713	7713	7713	7713	7713	0	0	7713
below or equ...	25	25	25	25	25	25	25	0	0	25
62607.0 - 73...	1880	1880	1880	1880	1880	1880	1880	0	0	1880
52172.5 - 62...	2454	2454	2454	2454	2454	2454	2454	0	0	2454
41738.0 - 52...	6358	6358	6358	6358	6358	4603	4603	1755	237	6595
31303.5 - 41...	1724	1724	1724	1724	1724	1346	1346	378	0	1724
83476.0 - 93...	64	64	64	64	64	64	64	0	0	64
73041.5 - 83...	2902	2902	2902	2902	2902	2902	2902	0	0	2902

Figure 6.6: Move statistics visualization: (1) table configuration; (2) statistics table; (3) bar chart visualizing a table row



Control-flow compliance alignment for Anonymous log imported from purchasing.xes on PO_release 1 skipped_w data

Attribute filter: All attributes

- Activity InvoiceReceipt is performed with incorrect PO_AMOUNT (711 violations)
  - Activity InvoiceReceipt is performed with incorrect PO_AMOUNT, when MATERIAL = material607 (711 violations)
  - Activity InvoiceReceipt is performed with incorrect PO_AMOUNT, when PO_AMOUNT in below or equal to 200000.0 (711 violations)
  - Activity InvoiceReceipt is performed with incorrect PO_AMOUNT, when ENTRY_AMOUNT in 41738.0 - 52172.5 (585 violations)
  - Activity InvoiceReceipt is performed with incorrect PO_AMOUNT, when USER_ID = resource120 (711 violations)
  - Activity InvoiceReceipt is performed with incorrect PO_AMOUNT, when ENTRY_AMOUNT in 31303.5 - 41738.0 (126 violations)
- GoodsReceipt (711 violations)
  - Activity GoodsReceipt is performed with incorrect PO_AMOUNT (711 violations)
- OutgoingPayment (711 violations)
  - Activity OutgoingPayment is performed with incorrect PO_AMOUNT (711 violations)
- When USER_ID = resource80 (711 violations)
- When USER_ID = resource121 (711 violations)
- When USER_ID = resource120 (711 violations)
  - When USER_ID = resource120, Activity InvoiceReceipt is performed with incorrect PO_AMOUNT (711 violations)
- PORelease1 (237 violations)
  - Activity PORelease1 is skipped (237 violations)
    - Activity PORelease1 is skipped, when ENTRY_AMOUNT in 41738.0 - 52172.5 (237 violations)
    - Activity PORelease1 is skipped, when MATERIAL = material27 (237 violations)
    - Activity PORelease1 is skipped, when USER_ID = resource28 (237 violations)
    - Activity PORelease1 is skipped, when PO_AMOUNT in 200000.0 and above (237 violations)
- When ENTRY_AMOUNT in 31303.5 - 41738.0 (378 violations)

Minimal confidence (CPIR) 0.0 — 0

Maximal confidence (CPIR) 1.0 — 1

Importance of activities and attributes

Activity	Importance (between 0.0 and 1.0)
ClearingCredit	0.5
ClearingDebit	0.5
GoodsReceipt	0.5
GoodsReceipt1	0.5
InvoiceReceipt	
InvoiceReceipt1	
Omega	0.5
OutgoingPayment	0.5
OutgoingPayment1	0.5

Attribute	Importance (between 0.0 and 1.0)
ENTRY_AMOUNT	0.5
MATERIAL	0.5
PO_AMOUNT	0.5
USER_ID	0.5

Figure 6.7: Problem report: (1) Problem List; (2) list configuration

6.2 Problem insight plugin

This section describes the plugin implementing the second step of the approach described in Chapter 5.

Table 6.2: ‘Get problem insight’ inputs and outputs

Plug-in Name	Get Problem insight
Input	ComplianceDashboard object
Output	InstanceClassifier object

The plugin uses as input a ’ComplianceDashboard’ object, which is produced by the ‘Get compliance dashboard’ plugin described in the previous section. This object contains both *Problem List* and *Data-Aware Replay Result* required by the plugin to run.

6.2.1 Integration into the Existing ProM Environment

In this section we discuss interactions with existing software required for the plugin to be executed. Figure 6.8 answers this question schematically.

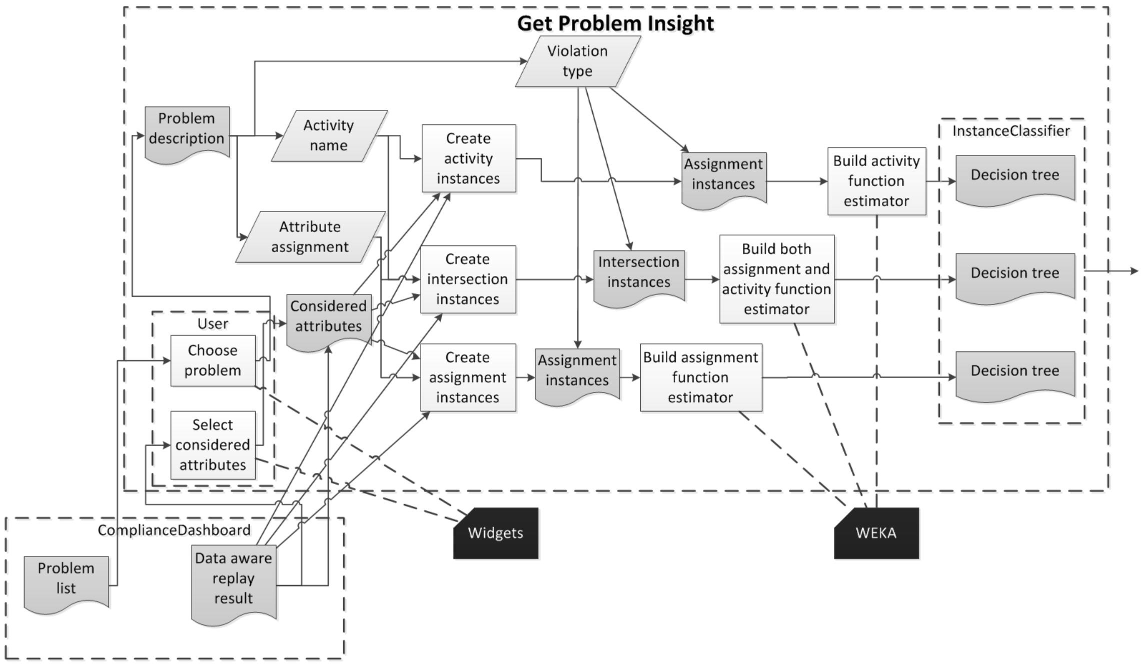


Figure 6.8: Interactions of ‘Get problem insight’ plugin

As shown in the figure, the input is provided in the form of ComplianceDashboard object. The plugin requires that the user chooses a problem from the list and attributes to consider during *building the function estimator* based on the attributes present in the analyzed event log (and hence the Data-Aware Replay Result). The user interface uses some of the interface elements implemented inside the Widgets package. The problem description is decomposed into three parts: Activity name, Attribute assignment, and Violation type (attribute assignment may be missing, depending on the selected problem).

Next, the plugin creates up to three groups of instances from the Data-Aware Replay Result, using *considered attributes*:

1. Instances based on Data-Aware Steps bearing the same activity name as specified in the selected problem. Assume that our problem is related to activity named *aname*. Then this group would include all steps from our Data-Aware Replay Result that have the activity name *aname*.

2. Instances based on Data-Aware Steps that have the same value of the attribute, as specified in the selected problem. Assume that our problem is related to attribute assignment (v, u), where $v \in V_L$ is a log attribute and $u \in D_v$ is its value. Then this group would include all steps from our Data-Aware Replay Result that assign value u for attribute v.

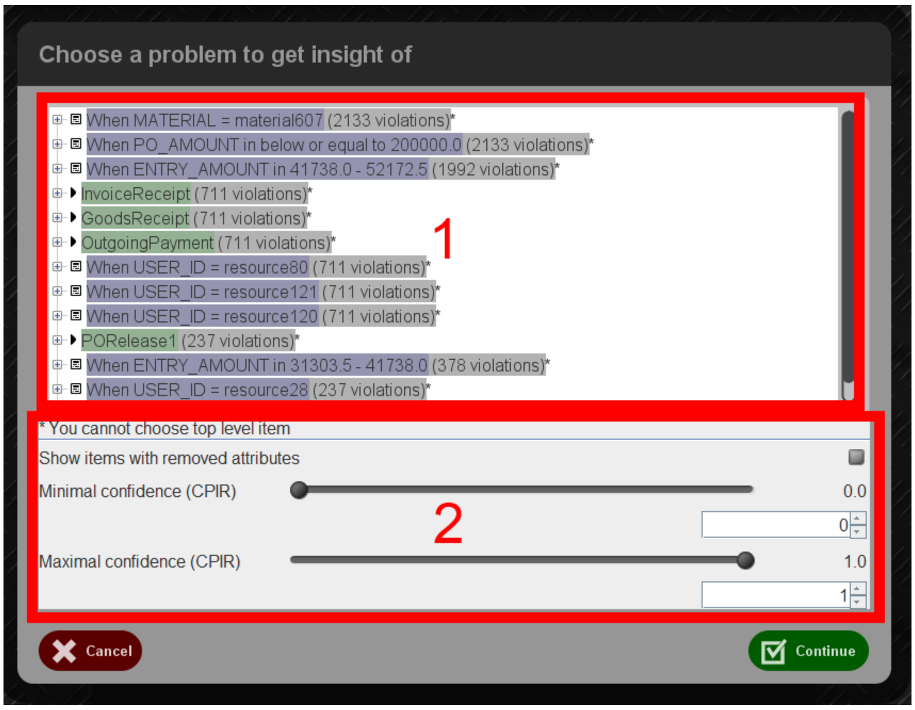
3. Instances based on Data-Aware Steps that have both the same attribute assignment and activity name, as specified in the selected problem. Basically, an intersection of sets 1 and 2.

Finally, the obtained groups of instances are used to create a function estimator for each of the groups. This is done using C4.5 classification tree, implemented within Weka[4]. The resulting trees are stored within the InstanceClassifier object, which is provided to the ProM framework as an output.

6.2.2 Graphical User Interface

Running the Plugin

After starting the plugin, the user is presented a deviation report similar to the visualization of Compliance Dashboard. The user has to select a single node, covering the problem of interest (Figure 6.9). Since the node has to correspond to a certain problem, the user is unable to choose a top level node. It is also possible to manipulate the tree with the confidence threshold slider in the same way as it is described in Section 6.1.2. The problem is used to select Data-Aware Steps which will be included into the instance set to run a classification on.



Choose a problem to get insight of

When MATERIAL = material607 (2133 violations)*
When PO_AMOUNT in below or equal to 200000.0 (2133 violations)*
When ENTRY_AMOUNT in 41738.0 - 52172.5 (1992 violations)*
InvoiceReceipt (711 violations)*
GoodsReceipt (711 violations)*
OutgoingPayment (711 violations)*
When USER_ID = resource80 (711 violations)*
When USER_ID = resource121 (711 violations)*
When USER_ID = resource120 (711 violations)*
PORelease1 (237 violations)*
When ENTRY_AMOUNT in 31303.5 - 41738.0 (378 violations)*
When USER_ID = resource28 (237 violations)*

* You cannot choose top level item

Show items with removed attributes

Minimal confidence (CPIR) 0.0 0

Maximal confidence (CPIR) 1.0 1

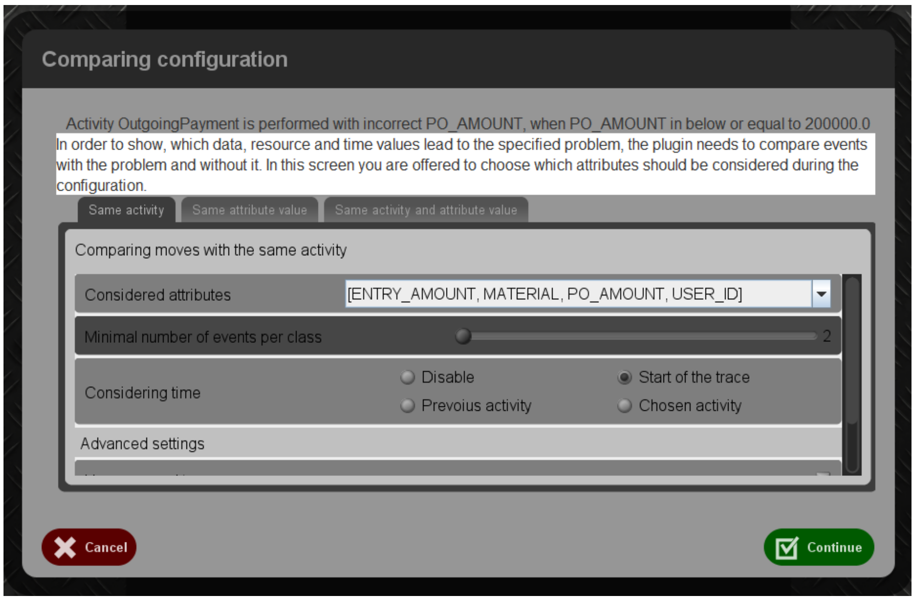
Figure 6.9: Choosing a problem for further investigation: (1) Problem list; (2) list configuration

[4]Weka is an open source suite for machine learning (`http://www.cs.waikato.ac.nz/ml/weka/`)

After choosing the problem, the user is offered to configure the classification tree (Figure 6.10). The plugin will offer to choose attributes, which should be considered during the comparing of problematic events and non-problematic ones. The user has to configure the way the Data-Aware Steps within instance groups are compared separately for each group. They should specify attributes that are considered during comparison and configure the classification problem. The plugin offers to consider the time of a Data-Aware Step occurrence. It can be considered in one of the following ways:

- not considered,
- time between the instance step and the start of the case,
- time between the instance step and the previous step or
- time between the instance step and the latest occurrence of a step with a specified activity name.

If the user decides to use the last option, they are offered to select the activity name using a combobox.


Comparing configuration

Activity OutgoingPayment is performed with incorrect PO_AMOUNT, when PO_AMOUNT in below or equal to 200000.0
In order to show, which data, resource and time values lead to the specified problem, the plugin needs to compare events with the problem and without it. In this screen you are offered to choose which attributes should be considered during the configuration.

Same activity | Same attribute value | Same activity and attribute value

Comparing moves with the same activity

Considered attributes: [ENTRY_AMOUNT, MATERIAL, PO_AMOUNT, USER_ID]

Minimal number of events per class: 2

Considering time: Disable [✓ Start of the trace] Prevoius activity Chosen activity

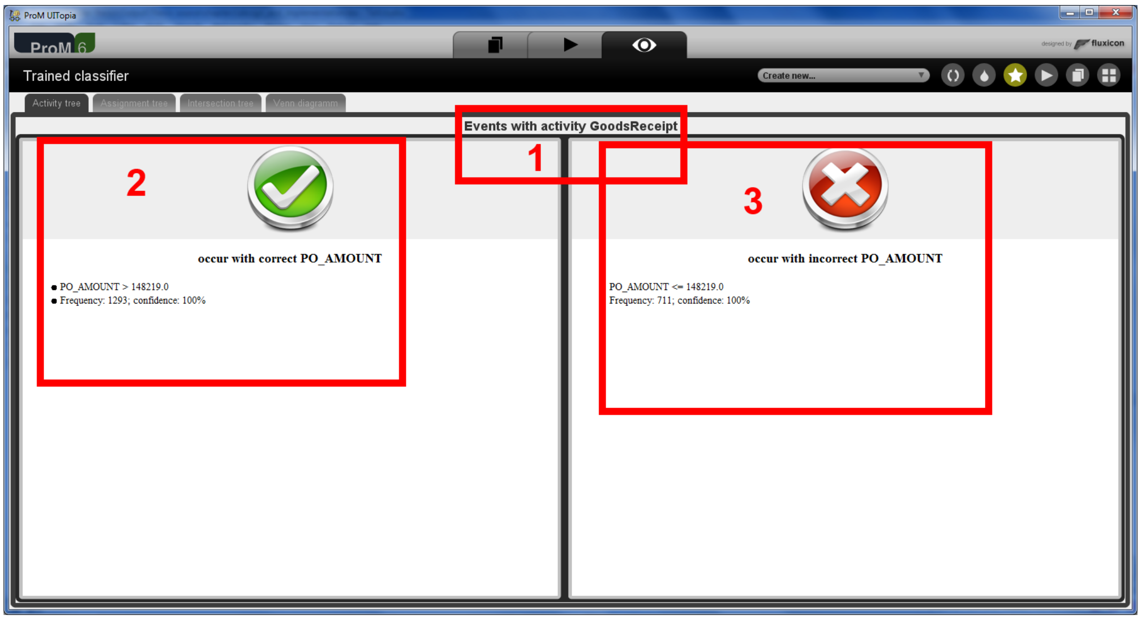
Advanced settings

Figure 6.10: Tree configuration

Plugin Output Visualization

The visualization (Figure 6.11) will list both the preconditions which lead to the presence of the specified problem and to its absence. The preconditions will be listed separately for events with specified activity, events with the specified attribute assignment and events that have both.

Another way of visualization (Figure 6.12) shows the preconditions attached to the corresponding parts of the Venn diagram, where the red areas correspond to the problematic parts and the green areas correspond to the non-problematic parts.



Events with activity GoodsReceipt

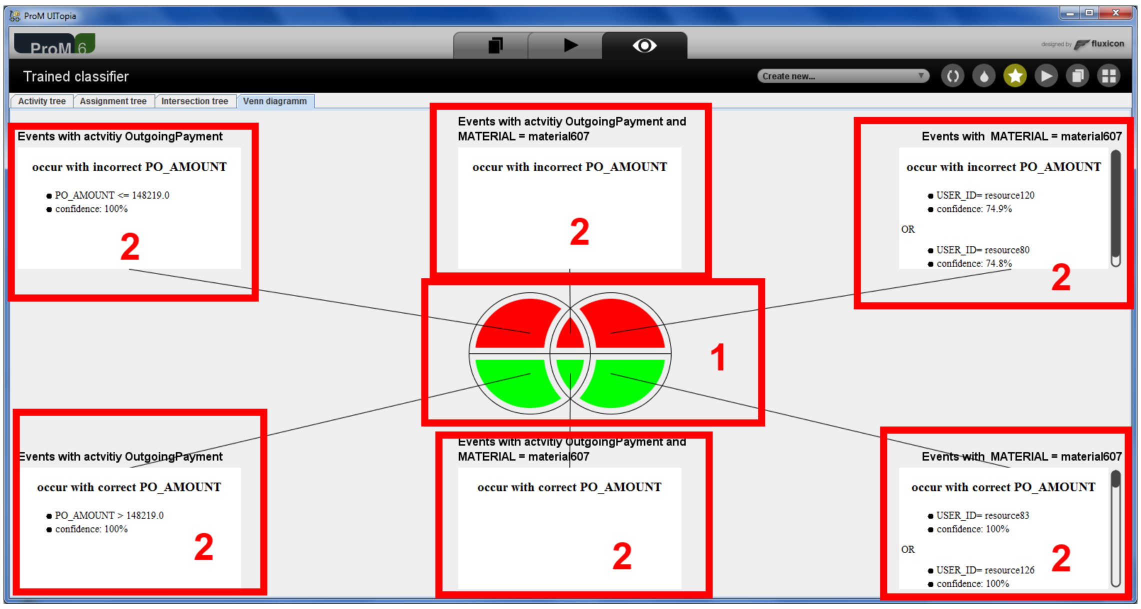
occur with correct PO_AMOUNT

- PO_AMOUNT > 148219.0
- Frequency: 1293; confidence: 100%

occur with incorrect PO_AMOUNT

PO_AMOUNT <= 148219.0
Frequency: 711; confidence: 100%

Figure 6.11: Tree visualization: (1) instances description; (2) root cause analysis of the instances without the problem; (3) analysis of the instances with the problem



Events with actvitiy OutgoingPayment

occur with incorrect PO_AMOUNT

- PO_AMOUNT <= 148219.0
- confidence: 100%

Events with actvitiy OutgoingPayment and MATERIAL = material607

occur with incorrect PO_AMOUNT

Events with MATERIAL = material607

occur with incorrect PO_AMOUNT

- USER_ID= resource120
- confidence: 74.9%

OR

- USER_ID= resource80
- confidence: 74.8%

Events with actvitiy OutgoingPayment

occur with correct PO_AMOUNT

- PO_AMOUNT > 148219.0
- confidence: 100%

Events with actvitiy OutgoingPayment and MATERIAL = material607

occur with correct PO_AMOUNT

Events with MATERIAL = material607

occur with correct PO_AMOUNT

- USER_ID= resource83
- confidence: 100%

OR

- USER_ID= resource126
- confidence: 100%

Figure 6.12: Venn diagram: (1) instance sets visualization; (2) root cause analysis of the sets

Chapter 7

Evaluation

In Chapter 6, we described how the approach proposed in this thesis was implemented. In this chapter, we describe the evaluation of the approach.

The evaluation serves multiple purposes. First, the correctness of the approach is verified using an artificial log with inserted violations. In this way, we make sure that we are able to judge whether the approach is indeed able to detect the violations and identify related diagnostics. Second, we address one of the objectives set in Chapter 1: *visualizing obtained diagnostics in a way understandable for users without a technical background.* Here, we verify that this objective is fulfilled by asking users to answer a questionnaire regarding violations in the analyzed event log. Third, we evaluate whether the developed approach is applicable for real-life scenarios by using a real-life dataset.

7.1 Evaluation using Artificial Data

In this section, we describe the evaluation of the developed approach using artificial data. To ensure that violations can be identified by the implemented software, an event log was adapted by inserting several violations. In order to see whether the provided output is really understandable, a user evaluation was performed. Users were asked to fill in a questionnaire, with questions regarding the output. In the latter study we would like to check whether the output of analysis (diagnostics) are visualized in a way which is comprehensible by users with less technical knowledge.

This section is organized as follows. Section 7.1.1 introduces the event log, the attributes it contains, the underlying process, and the inserted violations. Section 7.1.2 describes application of the proposed approach to the created log. Finally, Section 7.1.3 describes the results of the evaluation with users.

7.1.1 Log Description

This section describes an artificially generated event log that was created in order to perform the evaluation. It was generated using CPN tools[1]. This toolset is developed to create and record behavior of Petri nets. We created a model and recorded its executions. The purpose was to create a process as similar to a real one as possible.

Underlying process

A process model in Petri Net notation is shown in Figure 7.1. The model describes an example procurement process in a company, expressed as a sequence of events related to a single purchase entry. Please note that the figure shows only the control-flow of the process. Each entry corresponds to a certain number of units of some purchased product. A purchase order can have multiple

[1]http://cpntools.org/

entries. Also, there is an amount of money corresponding to each entry, which is the total cost of the purchased product.

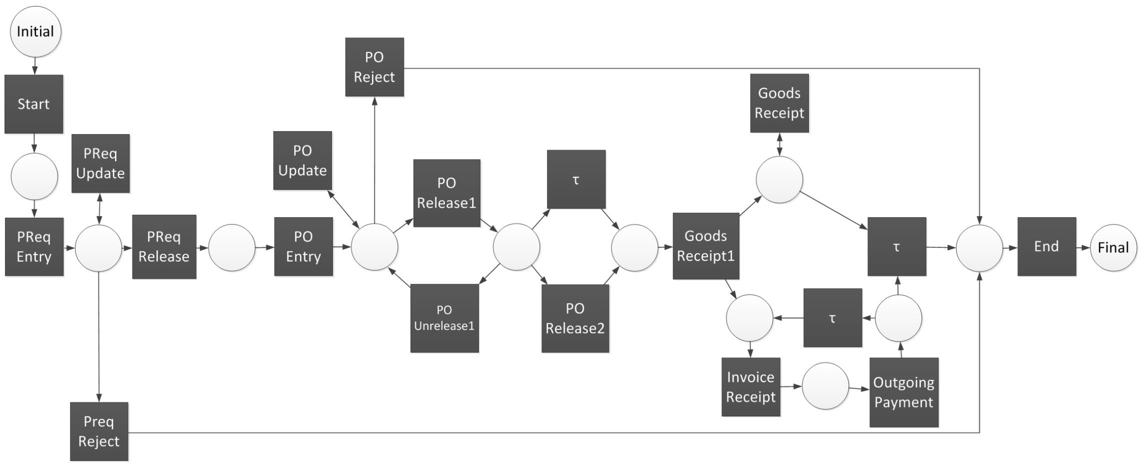


Figure 7.1: Artificial process

The process starts with a purchase requisition entry (*PReqEntry*), which may be updated (*PReqUpdate*) an arbitrary number of times. After that, the requisition is either rejected (*PReqReject*) or released (*PReqRelease*). In the former case the process instance ends. If the requisition is released, a purchase order is created based on it. Similarly to the requisition, the purchase order can be updated (*POUpdate*) several times and rejected (*POReject*). An approval of a purchase order consists of one step (*PORelease1*) for orders with amounts below €200,000 and of two steps (*PORelease1* and *PORelease2*) done by separate persons for orders with amounts above €200,000. Also, the release can be reverted after the first step, after which the order has to be either released again or rejected. If it is rejected, the case ends.

After all approvals are passed, the supplier sends the goods, and their receipt (*GoodsReceipt1*) is recorded in the system. Each purchase order has to have at least one Goods Receipt (unless it was rejected). Then, the Invoice Receipt (*InvoiceReceipt*) occurs and is paid (*OutgoingPayment*) afterwards. If the amount of purchase order is above €200,000, multiple Goods Receipts (*GoodsReceipt*) as well as sequences of Invoice Receipts (*InvoiceReceipt*) and payments (*OutgoingPayment*) occur. After all goods related to a purchase order are received and respective payments are completed, the case finishes.

Table 7.1: The Artificial Log attributes

Attributes with fixed values within case	
Name	**Description**
ENTRY_AMOUNT	Total amount of the given purchase order entry, i.e. entry to which the case corresponds
PO_AMOUNT	Total amount of a purchase order, which contains the given entry
MATERIAL	Product, which is being purchased
Attributes with values that may change as the case progresses	
Name	**Description**
concept:name	Name of the event
time:timestamp	Timestamp of the event
USER_ID	ID of the user who is responsible for event

Log attributes

Each event has a number of attributes. Some of them are unchanged within single trace: MATERIAL, which is the purchased item name; ENTRY_AMOUNT, which is the amount of money corresponding to the current entry and PO_AMOUNT, which is the amount of money corresponding to the entire purchase order. The others are individual for each event: USER_ID, which is the system name of the user responsible for the event; event name and timestamp. All attributes are summarized in Table 7.1.

Table 7.2: Artificial log violations and underlying context

Rule #	Violated rule	# of cases/ events	Attributes	Values
1	All orders (except rejected) are a subject to at least a single PORelease.	237/237	Material	material607
			PO_AMOUNT	€0 - €200,000
			ENTRY_AMOUNT	€40,000 - €50,000
2	GoodsReceipt (GR), InvoiceReceipt (IR) and OutgoingPayment (Payment) are allowed to occur multiple times only for cases with a total amount of purchase order above €200,000	237/711	Material	material607
			USER_ID for GR	resource80
			USER_ID for IR	resource120
			USER_ID for Payment	resource121
			PO_AMOUNT	€0 - €200,000
			ENTRY_AMOUNT	€40,000 - €50,000
3	Activity GoodsReceipt (GR) can be performed only by the users with IDs $resource71, resource72, \ldots, resource100$	237/608	Material	material611, material612, material613, material614, material615
			USER_ID for GR	resource125, resource126, resource127
			ENTRY_AMOUNT	€0 - €10,000
4	Invoice cannot be paid (OutgoingPayment) by the same user who received it (InvoiceReceipt), though both OutgoingPayment and InvoiceReceipt are performed by users from the same department with the same responsibilities.	237/237	USER_ID for Payment	resource128, resource129
			ENTRY_AMOUNT	€55,000 - €65,000
			Time between IR and Payment	0 - 0.5hr

Inserted violations

Several cases with violations were inserted into the event log in order to verify the implementations ability to identify the violations and the underlying context. The related information is summarized in Table 7.2. The table lists the 4 rules that were checked during the evaluation. For each of the rules, the specified number of cases were added to the log with violating events. These events have the attribute values listed in the table. For the attributes with a range or a list of values, the values are distributed randomly.

The first rule states that activity *PORelease1* must be executed within each case, unless *PReqReject* or *POReject* occur. As seen in Table 7.2, 237 cases were added to the log, where neither any of the rejects is present, nor *PORelease1* is present. Since the rule restricts only a single occurrence of the activity, 237 moves on model for the activity *PORelease1* should be detected.

According to the next rule, the activities *GoodsReceipt, InvoiceReceipt and OutgoingPayment* are allowed to occur more than once only for purchase orders with amount (PO_AMOUNT) above €200,000. There are 237 cases in the log where PO_AMOUNT is below €200,000 while all of the mentioned activities are performed more than once (totally 711 times each).

Another rule restricts users who are allowed to accept purchased products to users with specific IDs. These users are assumed to belong to a single department. There are 237 cases where this operation was performed by non-eligible users (namely, *resource125, resource126*, and *resource127*).

The final rule is an example of segregation of duties.The rule is violated 237 times within 237 cases, i.e. *OutgoingPayment* was performed by the same person as *InvoiceReceipt.*

7.1.2 Applying the Compliance Framework

In this section, we describe obtaining diagnostic information about the inserted violations using the developed plugin.

Scenario 1

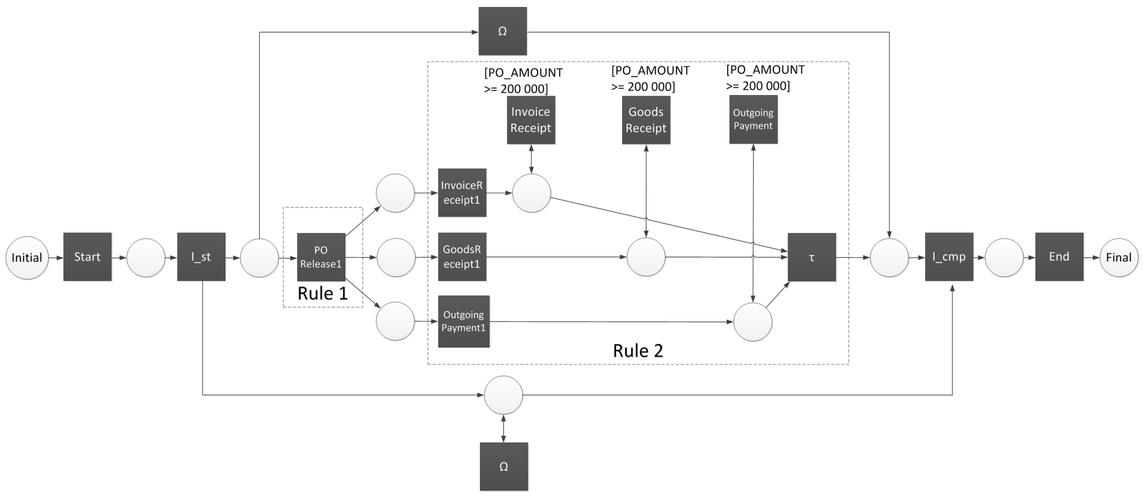
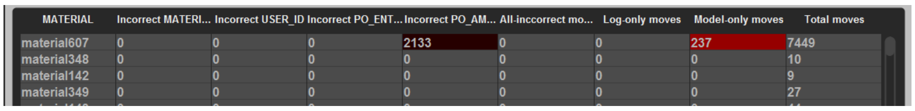


Figure 7.2: Rules 1 and 2 specified in the form of a Petri Net with Data

The rule, specified as a Petri Net with Data corresponding to this scenario is shown in Figure 7.2. This Petri Net expresses rules 1 and 2 from Table 7.2 and thus we should be able to discover the attribute values corresponding to violations of the mentioned rules.

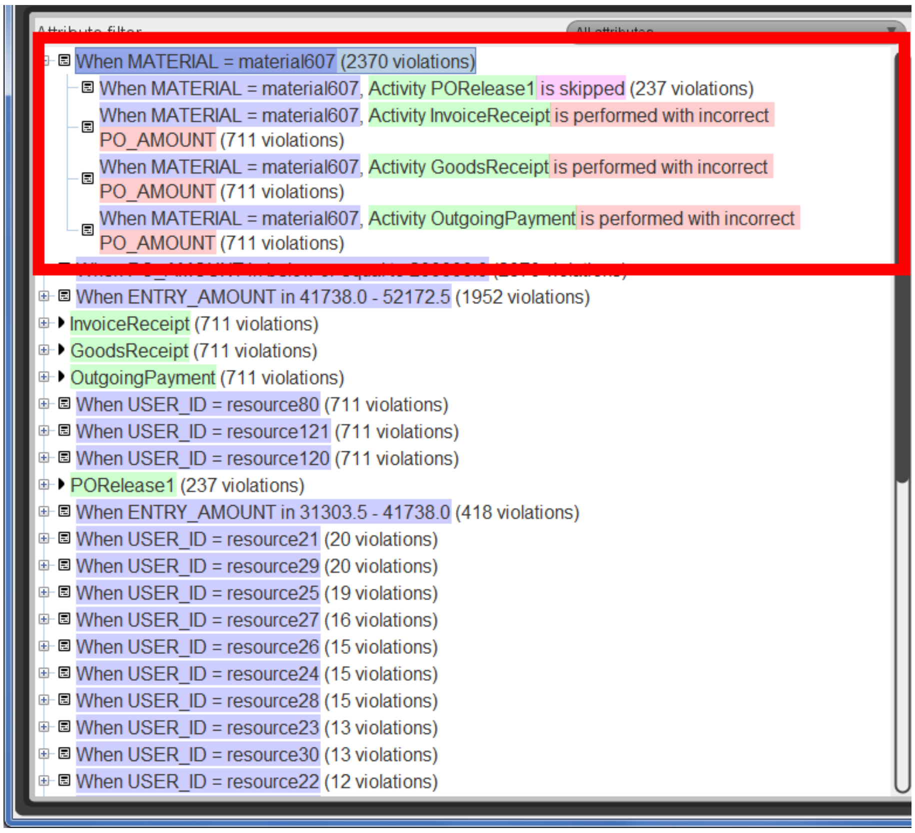
We are going to start the analysis by starting the 'Get Compliance Dashboard' plugin and examining the deviations statistics. The number of violations per MATERIAL value is shown in

Figure 7.3. According to the figure, the checked rule is violated only when the MATERIAL value is *material607* and the number of violations corresponds to that given in Table 7.2.



MATERIAL	Incorrect MATERI...	Incorrect USER_ID	Incorrect PO_ENT...	Incorrect PO_AM...	All-inccorrect mo...	Log-only moves	Model-only moves	Total moves
material607	0	0	0	2133	0	0	237	7449
material348	0	0	0	0	0	0	0	10
material142	0	0	0	0	0	0	0	9
material349	0	0	0	0	0	0	0	27

Figure 7.3: Number of violations per MATERIAL value (violation occur only for events with MATERIAL = *material607*)



Attribute filter
All attributes

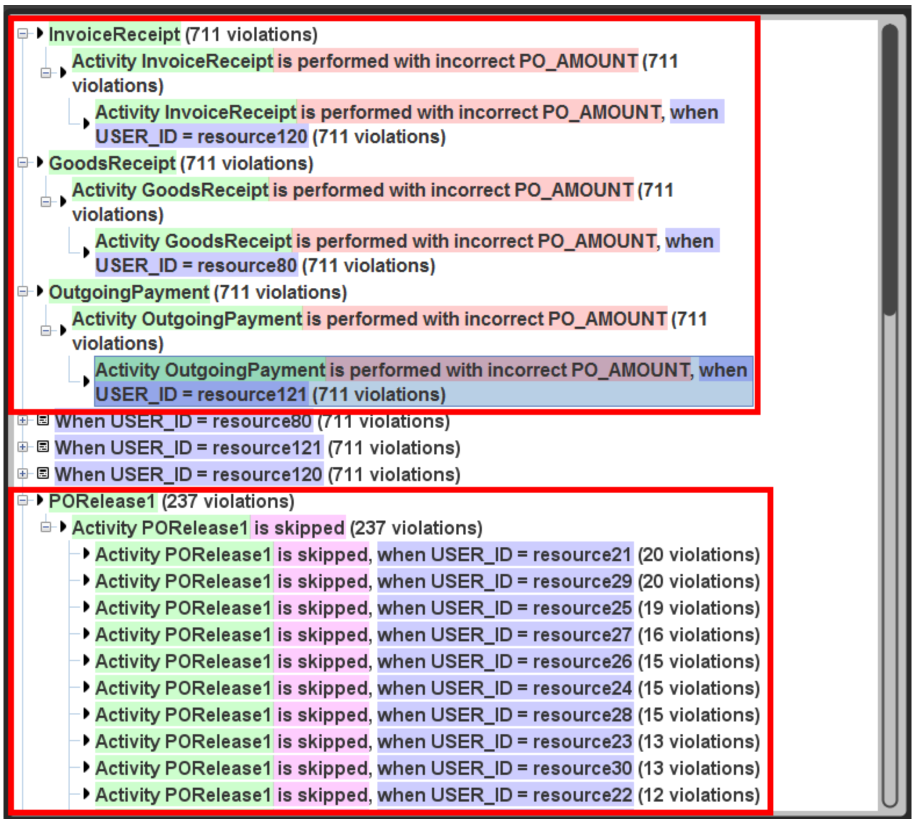
When MATERIAL = material607 (2370 violations)
- When MATERIAL = material607, Activity PORelease1 is skipped (237 violations)
- When MATERIAL = material607, Activity InvoiceReceipt is performed with incorrect PO_AMOUNT (711 violations)
- When MATERIAL = material607, Activity GoodsReceipt is performed with incorrect PO_AMOUNT (711 violations)
- When MATERIAL = material607, Activity OutgoingPayment is performed with incorrect PO_AMOUNT (711 violations)

[illegible]
When ENTRY_AMOUNT in 41738.0 - 52172.5 (1952 violations)
InvoiceReceipt (711 violations)
GoodsReceipt (711 violations)
OutgoingPayment (711 violations)
When USER_ID = resource80 (711 violations)
When USER_ID = resource121 (711 violations)
When USER_ID = resource120 (711 violations)
PORelease1 (237 violations)
When ENTRY_AMOUNT in 31303.5 - 41738.0 (418 violations)
When USER_ID = resource21 (20 violations)
When USER_ID = resource29 (20 violations)
When USER_ID = resource25 (19 violations)
When USER_ID = resource27 (16 violations)
When USER_ID = resource26 (15 violations)
When USER_ID = resource24 (15 violations)
When USER_ID = resource28 (15 violations)
When USER_ID = resource23 (13 violations)
When USER_ID = resource30 (13 violations)
When USER_ID = resource22 (12 violations)

Figure 7.4: Violations corresponding to material607 (*PORelease1* is skipped and *GoodsReceipt*, *InvoiceReceipt*, and *OutgoingPayment* are performed with incorrect PO_AMOUNT)

If we switch to the deviations report, we can see, what are the exact violations, in which material607 is involved (Figure 7.4). As shown in the figure, the value occurs in violations of both rules: *PORelease1* is skipped (violation of the rule 1) and *GoodsReceipt*, *InvoiceReceipt*, and *OutgoingPayment* are performed with incorrect PO_AMOUNT (violation of the rule 2). The report also shows users who violate the rule 2 by performing the activities with incorrect PO_AMOUNT (Figure 7.5). The discovered context matches the information given in Table 7.2. The USER_ID information for *PORelease1* violations is irrelevant, since the event is actually skipped, thus there is no way to find out the responsible person with only event log information. It could be that there is a dependency between a person who performed the step previous to the violation and the violation itself, but there is no such relation, since different users are uniformly distributed in this artificial dataset (Figure 7.5).

Finally, in order to identify the ENTRY_AMOUNT, we run the 'Get Problem Insight' plugin on the item 'Activity OutgoingPayment is performed with incorrect PO_AMOUNT'. As it is shown



InvoiceReceipt (711 violations)
Activity InvoiceReceipt is performed with incorrect PO_AMOUNT (711 violations)
Activity InvoiceReceipt is performed with incorrect PO_AMOUNT, when USER_ID = resource120 (711 violations)
GoodsReceipt (711 violations)
Activity GoodsReceipt is performed with incorrect PO_AMOUNT (711 violations)
Activity GoodsReceipt is performed with incorrect PO_AMOUNT, when USER_ID = resource80 (711 violations)
OutgoingPayment (711 violations)
Activity OutgoingPayment is performed with incorrect PO_AMOUNT (711 violations)
Activity OutgoingPayment is performed with incorrect PO_AMOUNT, when USER_ID = resource121 (711 violations)
When USER_ID = resource80 (711 violations)
When USER_ID = resource121 (711 violations)
When USER_ID = resource120 (711 violations)
PORelease1 (237 violations)
Activity PORelease1 is skipped (237 violations)
Activity PORelease1 is skipped, when USER_ID = resource21 (20 violations)
Activity PORelease1 is skipped, when USER_ID = resource29 (20 violations)
Activity PORelease1 is skipped, when USER_ID = resource25 (19 violations)
Activity PORelease1 is skipped, when USER_ID = resource27 (16 violations)
Activity PORelease1 is skipped, when USER_ID = resource26 (15 violations)
Activity PORelease1 is skipped, when USER_ID = resource24 (15 violations)
Activity PORelease1 is skipped, when USER_ID = resource28 (15 violations)
Activity PORelease1 is skipped, when USER_ID = resource23 (13 violations)
Activity PORelease1 is skipped, when USER_ID = resource30 (13 violations)
Activity PORelease1 is skipped, when USER_ID = resource22 (12 violations)

Figure 7.5: Top: users, involved in violation of rule 2 (*resource120, resource80, resource121*). Bottom: users who executed the activity previous to PORelease1 in violating cases

in Figure 7.6, the bounds are €39,955.0 - €49,999.0, which (approximately) matches the data in Table 7.2.

Scenario 2

The rule, specified as a Petri Net with Data corresponding to this scenario is shown in Figure 7.7. This Petri Net expresses rule 3 from Table 7.2.

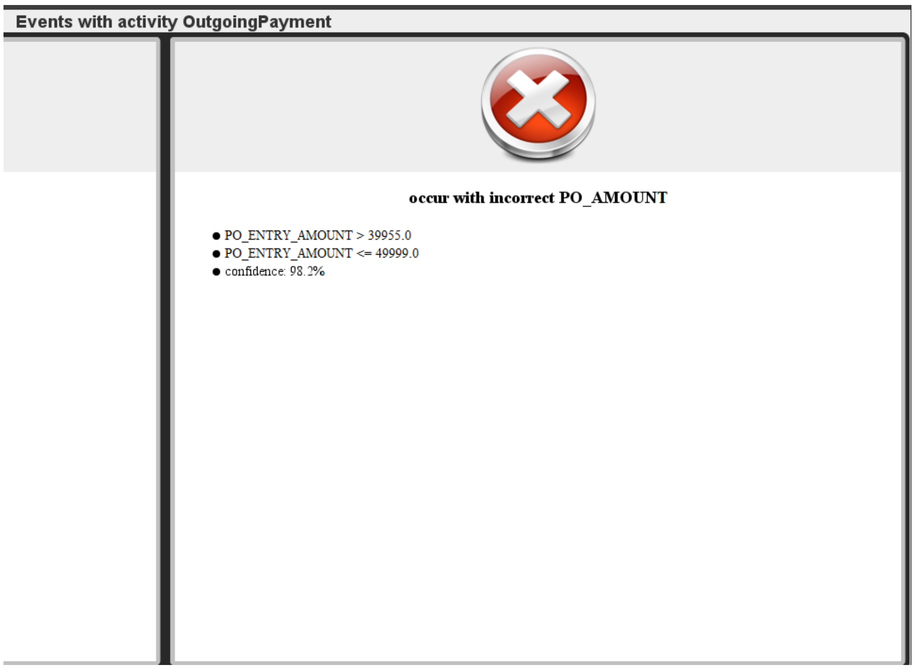
In Figure 7.8 it is shown that activity GoodsReceipt occurs 608 times in moves with incorrect USER_ID. The report in Figure 7.9 provides additional insight into the problem: the ENTRY_AMOUNT interval is €0.0 - €10,434.5. It is also clear that the expected number of 608 violating steps are detected.

In order to identify which users are responsible for the violations, the 'Get problem insight' plugin is run on the item 'Activity GoodsReceipt is performed with incorrect USER_ID', whose output confirms that all the responsible users are identified, namely *resource125, reosurce126*, and *resource127* (Figure 7.10). The same is true for MATERIAL attribute, which is shown in Figure 7.11.

Scenario 3

This scenario focuses on checking rule 4 from Table 7.2. The rule, specified as a Petri Net with Data corresponding to this scenario, is shown in Figure 7.12.

By observing the violation report (Figure 7.13), we are able to identify that users *resource128* and *resource129* are responsible for the violations. It however does not show precise bounds for ENTRY_AMOUNT. We run the 'Get Problem Insight' plugin on the item 'Activity OutgoingPay-



Events with activity OutgoingPayment

occur with incorrect PO_AMOUNT

- PO_ENTRY_AMOUNT > 39955.0
- PO_ENTRY_AMOUNT <= 49999.0
- confidence: 98.2%

Figure 7.6: ENTRY_AMOUNT bounds (€39,955.0 - €49,999.0)

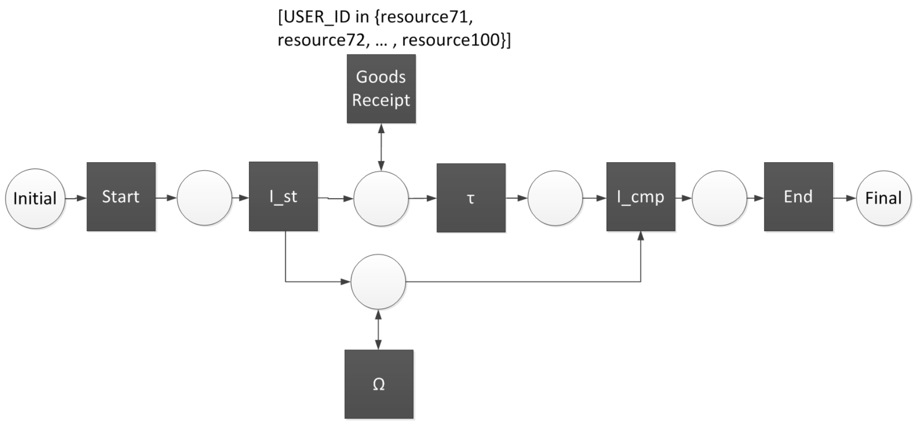
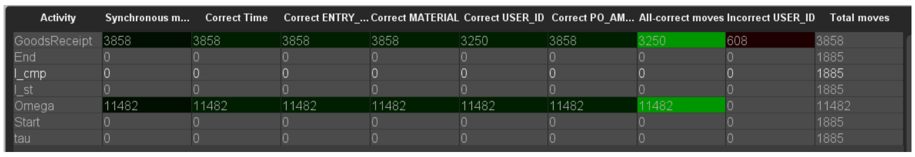


Figure 7.7: Rule 2 specified in Petri Net with Data

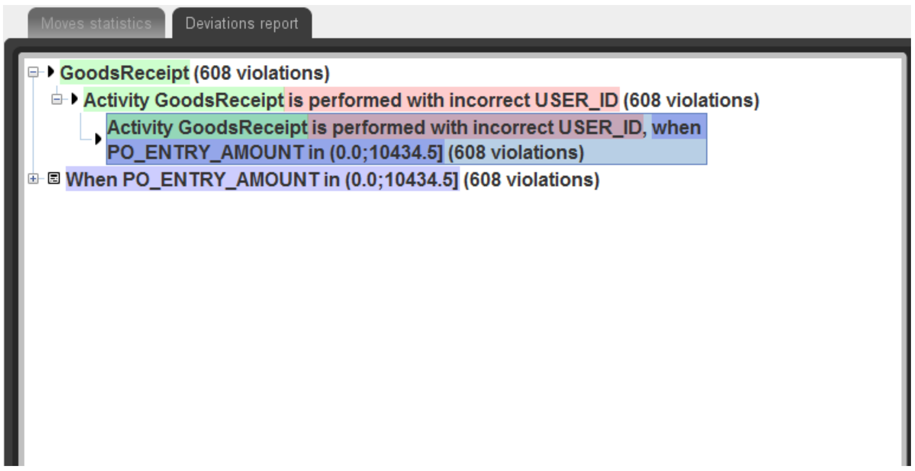


Activity	Synchronous m...	Correct Time	Correct ENTRY_...	Correct MATERIAL	Correct USER_ID	Correct PO_AM...	All-correct moves	Incorrect USER_ID	Total moves
GoodsReceipt	3858	3858	3858	3858	3250	3858	3250	608	3858
End	0	0	0	0	0	0	0	0	1885
I_cmp	0	0	0	0	0	0	0	0	1885
I_st	0	0	0	0	0	0	0	0	1885
Omega	11482	11482	11482	11482	11482	11482	11482	0	11482
Start	0	0	0	0	0	0	0	0	1885
tau	0	0	0	0	0	0	0	0	1885

Figure 7.8: Scenario 2 violation statistics: GoodsReceipt occurs 608 time with incorrect USER_ID

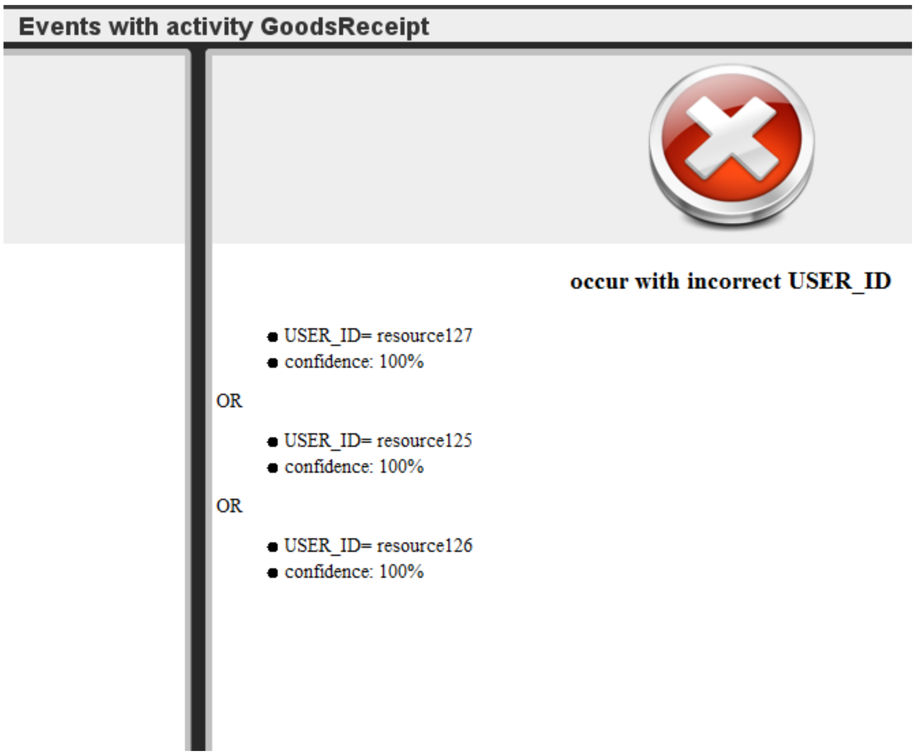
ment is performed with incorrect USER_ID' in order to discover them. This allows us to reveal that the bounds for the attribute are €54,855.0 - €56,735.5 and €56,905.0 - €64,866.0 (Figure 7.14).

Besides the data- and organizational-related context, the time-related context is also involved in the violations. It can be discovered by running the 'Get Problem Insight' plugin on the same item as previously, with the difference that we enable considering the time between *OutgoingPayment* and *InvoiceReceipt*. This allows us to identify that the usual time interval between them for



GoodsReceipt (608 violations)
Activity GoodsReceipt is performed with incorrect USER_ID (608 violations)
Activity GoodsReceipt is performed with incorrect USER_ID, when PO_ENTRY_AMOUNT in (0.0;10434.5] (608 violations)
When PO_ENTRY_AMOUNT in (0.0;10434.5] (608 violations)

Figure 7.9: ENTRY_AMOUNT involved in scenario 2 (€0.0 - €10,434.5)


Events with activity GoodsReceipt

occur with incorrect USER_ID

- USER_ID= resource127
- confidence: 100%

OR

- USER_ID= resource125
- confidence: 100%

OR

- USER_ID= resource126
- confidence: 100%

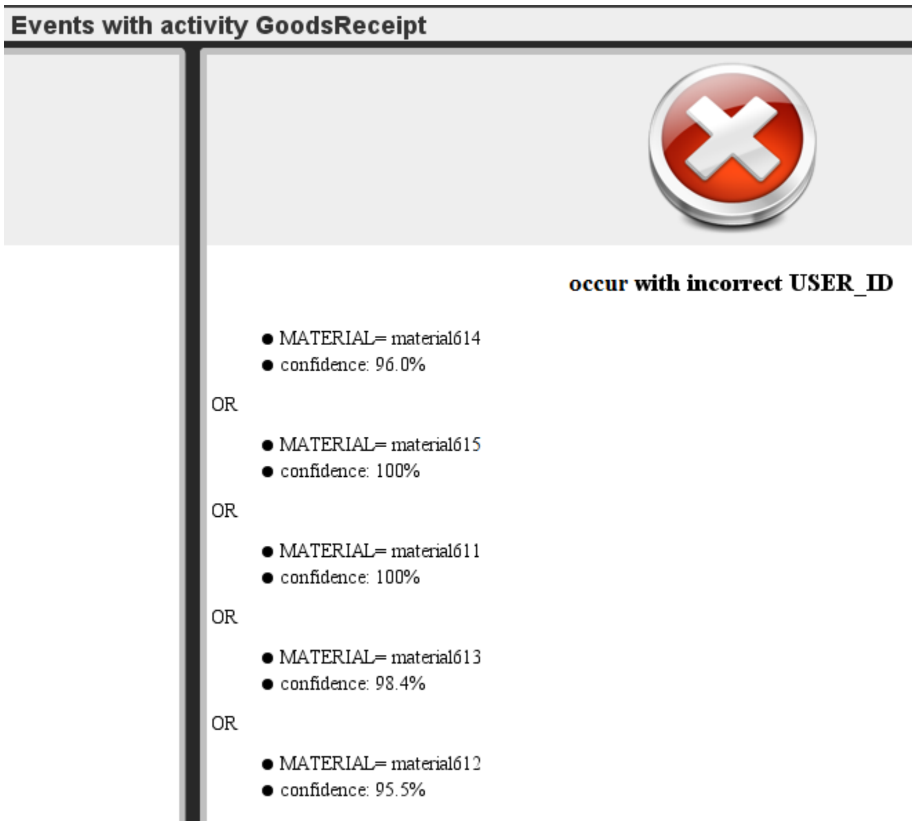
Figure 7.10: Users involved in scenario 2 (*resource125, reosurce126, resource127*)

violating cases is below 30 minutes (Figure 7.15).

As can be seen from the results presented, all the inserted context was discovered using the plugins.

7.1.3 User Evaluation

In this section, we aim to evaluate the understandability of the output of the implemented plugins for end users. In order to perform an evaluation, a number of workshops was held. The structure of these workshops was the following:


Events with activity GoodsReceipt

occur with incorrect USER_ID

- MATERIAL= material614
- confidence: 96.0%

OR

- MATERIAL= material615
- confidence: 100%

OR

- MATERIAL= material611
- confidence: 100%

OR

- MATERIAL= material613
- confidence: 98.4%

OR

- MATERIAL= material612
- confidence: 95.5%

Figure 7.11: MATERIAL values involved in scenario 2 (*material611,...,material615*)

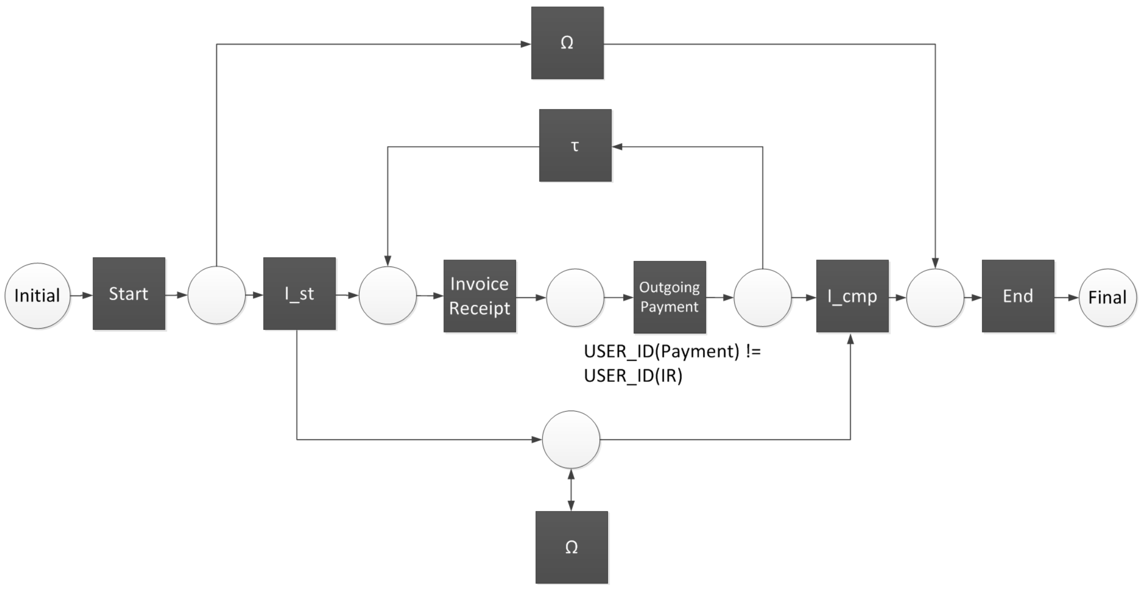
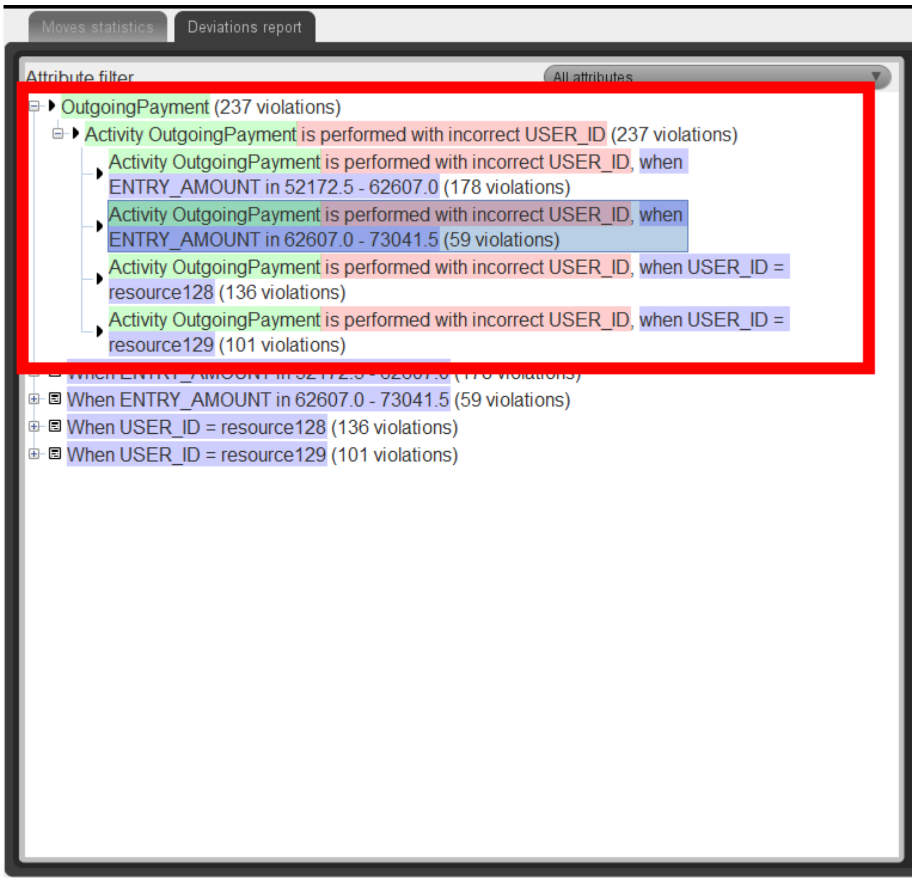


Figure 7.12: Rule 3 specified in Petri Net with Data

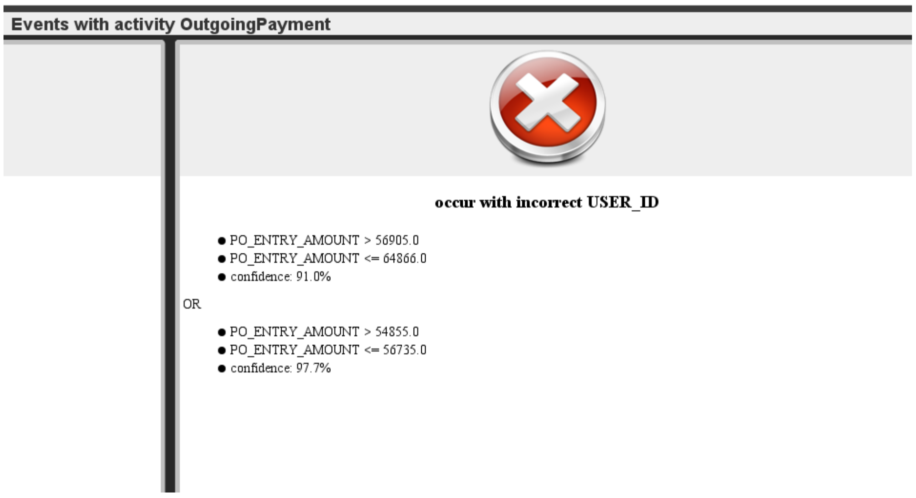
1. First, users were given a general presentation of the approach. This allowed to make them familiar with the key notions and understand how the approach works;

2. Second, users were given a demonstration of the developed plugins in order to familiarize



OutgoingPayment (237 violations)
Activity OutgoingPayment is performed with incorrect USER_ID (237 violations)
Activity OutgoingPayment is performed with incorrect USER_ID, when ENTRY_AMOUNT in 52172.5 - 62607.0 (178 violations)
Activity OutgoingPayment is performed with incorrect USER_ID, when ENTRY_AMOUNT in 62607.0 - 73041.5 (59 violations)
Activity OutgoingPayment is performed with incorrect USER_ID, when USER_ID = resource128 (136 violations)
Activity OutgoingPayment is performed with incorrect USER_ID, when USER_ID = resource129 (101 violations)
When ENTRY_AMOUNT in 52172.5 - 62607.0 (178 violations)
When ENTRY_AMOUNT in 62607.0 - 73041.5 (59 violations)
When USER_ID = resource128 (136 violations)
When USER_ID = resource129 (101 violations)

Figure 7.13: Users involved in scenario 3 (*resource128*, *resource129*)


Events with activity OutgoingPayment

occur with incorrect USER_ID

- PO_ENTRY_AMOUNT > 56905.0
- PO_ENTRY_AMOUNT <= 64866.0
- confidence: 91.0%

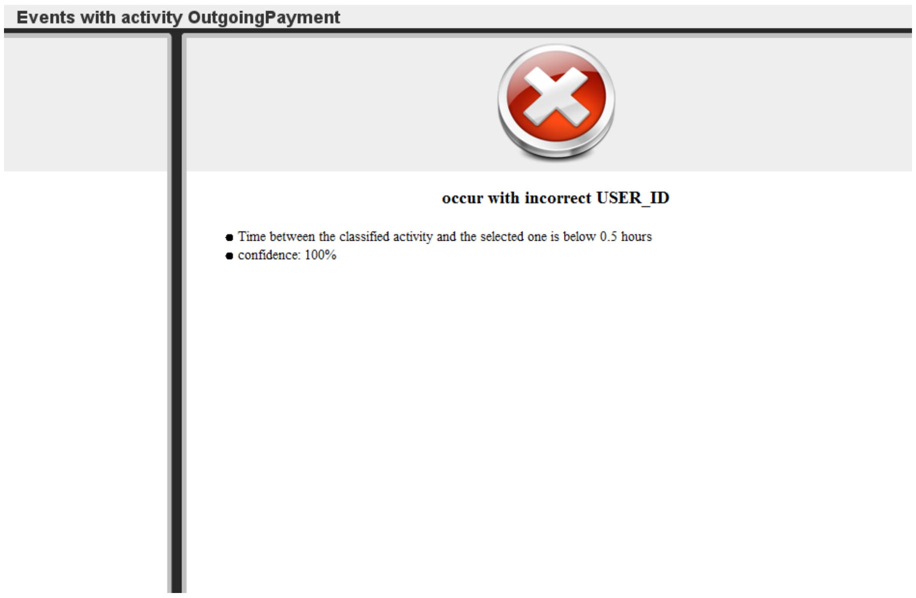
OR

- PO_ENTRY_AMOUNT > 54855.0
- PO_ENTRY_AMOUNT <= 56735.0
- confidence: 97.7%

Figure 7.14: ENTRY_AMOUNT involved in scenario 3 (€54,855.0 - €56,735.5 and €56,905.0 - €64,866.0)

them with the interface;

3. Third, users were asked to run scenarios based on the artificial data and answer a number of questions regarding the obtained output. The correctness of the answers would show their



occur with incorrect USER_ID

- Time between the classified activity and the selected one is below 0.5 hours
- confidence: 100%

Figure 7.15: Time context of scenario 3 (all the violations (confidence 100 %) occur when *OutgoingPayment* is performed within 30 minutes after *InvoiceReceipt*)

comprehension of the obtained results.

Alternatively, it was possible to do the evaluation remotely. All instructions and explanations were distributed to the users who preferred this format. The evaluation was performed with a group of 7 students at TU/e and with a group of 4 people working in assurance and forensic areas at PwC.

The questionnaire consisted of 10 questions, and was answered by 11 persons. In order to answer the questions, a user had to run both 'Get compliance dashboard' and 'Get problem insight' plugins for each scenario. The questionnaire is shown in Appendix B. Table 7.3 shows which questions were answered correctly.

As shown in the table, most mistakes among the TU/e students were made while answering questions 9 and 10. Both questions are related to output of the 'Get problem insight' plugin. Possibly, the reason is the wrong choice of considered attributes for the execution of the plugin. However, some students who were mistaken in question 9, provided the correct lower bound of ENTRY_AMOUNT values for the violating events, while the question asked to provide the upper bound.

The PwC users made most mistakes answering the question asking which attribute: PO_AMOUNT or ENTRY_AMOUNT had stronger relation to the problem in scenario 2. It is likely that the confusion was caused by the fact that some of the list items corresponding to the assignments of PO_AMOUNT had higher relevance values than that of ENTRY_AMOUNT items, while the correct answer was the latter.

Besides answering questions, users were given an opportunity to provide their comments regarding the developed plugins. Some of the comments criticized data representation. This included the way numerical ranges are expressed and the color scheme.

Another flaw, according to the feedback, is the interface. It was claimed that there is too much information on the screen and that it was hard to choose the correct attributes for the 'Get problem insight' plugin. Besides, it was mentioned that the metric of relevance between violation and context is not straightforward: increasing the relevance threshold does not guarantee that most relevant attribute assignments remain.

The users' input was used to improve the data representation and the interface. Such improvements include changing the way to express numerical ranges or showing IDs of the cases for each item in the problem list. Another improvement was to extend the user's capabilities in refining the list by adding a filter that allows them to keep only assignments of the selected attribute.

Despite the criticism, it was noted that the concept is promising and should be developed further. Also, as Table 7.3 shows, there are few incorrect answers, which indicates that the output is indeed comprehensible.

Table 7.3: Respondents' performance

Participant	**Affiliation**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**
1	TU/e	+	+	+	0	+	0	+	+	-	-
2	TU/e	+	+	+	0	+	0	+	+	-	-
3	PwC	+	+	+	-	+	-	+	+	+	+
4	TU/e	+	+	±	+	+	+	+	+	-	+
5	TU/e	+	+	±	+	+	+	+	+	-	+
6	PwC	+	+	+	+	+	-	+	+	+	+
7	PwC	+	+	+	+	+	-	+	+	+	+
8	TU/e	+	+	+	+	+	+	+	+	+	-
9	TU/e	+	+	+	+	+	+	+	+	+	+
10	TU/e	+	+	+	+	+	+	+	+	+	+
11	PwC	+	+	+	+	+	+	+	+	+	+

Legend: '+' – correct answer; '-' – incorrect answer; '±' – partially correct answer; '0' – omitted answer

7.2 Evaluation Using Real-Life Data (removed)

The section was removed due to confidentiality issues.

Chapter 8

Conclusions

8.1 Summary of Contributions

In this master thesis, we analyzed problems encountered when trying to obtain context-related information about violations discovered during compliance checking of event logs. These problems motivate the research goal stated in Chapter 1:

Develop a systematic approach to get insights about the violations detected during compliance checking through the analysis of context-related information available in the business process and present the obtained results in a way understandable for users not having deep technical knowledge.

Before proposing a new approach, we performed a study of the related work on each element in the research scope: compliance checking, identifying relations between context information and violations, visualizing results for users without a technical background. First, we selected approaches for performing compliance checking and used these to obtain violation data. We also identified a number of limitations in existing approaches. Knowledge of process mining is required in order to understand the way their output is expressed. Moreover, some of the approaches, focusing on context visualization, do not allow to explicitly distinguish violating events or cases. Finally, there is only one approach, introduced in [27] that allows to identify the relation between violations and the underlying context. However, our approach allows to find context patterns across different violations easier.

Our main contribution is an approach and tool to provide diagnostics of violations discovered during compliance checking, described in Chapter 4. To provide diagnostics we propose two phases. First, a summary of discovered violations is created. The summary consists of statistics and a list of problems. The list relates problems to the underlying context using $CPIR_{CF}$ measure, defined for this purpose. Each item in the list corresponds to some subset of the observed violations bearing a specific activity name, attribute assignment, and violation type. Moreover, Chapter 5 describes the way we discover the root cause of a specific problem, which is selected by the user. The discovery is based on classifying data tuples, built upon steps in the Data-Aware Replay Result. The visualization of the result includes describing features that distinguish the problematic instances from the non-problematic ones.

Another contribution is to visualize the obtained results in a way that does not require a technical background to be understandable. The visualization is described in Chapter 6. The data is visualized in different forms. Violation statistics are visualized in the form of a table and bar chart, where columns and bars are colored depending on whether they represent a number of violating or non-violating steps. The list of identified problems is represented as a hierarchical collection of statements, where each level in the hierarchy provides more details than the previous one. As for a problem's root cause, it is represented in two forms: first, as a list of conditions leading to either the occurrence, or non-occurrence of the problem and second, as a number of preconditions, attached to a certain part of a diagram. Each part of the diagram depicts a certain subset of the Data-Aware Steps in the log.

To verify our approach, we analyzed an artificially generated event log with purposefully inserted violations. Applying our approach allowed us to identify all the violations and underlying context. In order to verify that the output is comprehensible for business users, evaluation workshops were held, during which users answered a number of questions regarding the visualization. Correctness of a majority of the answers showed that the output is indeed understood.

8.2 Limitations and Future Work

To conclude, we discuss the limitations and future work with respect to the following aspects: (1) compliance checking and obtaining violation data (2) identifying the relation between context info and deviations, and (3) visualizing results for business users.

Some of the ideas described in this section are based on the feedback obtained during the evaluations. The limitations concern both the conceptual and the implementation level.

8.2.1 Compliance Checking and Obtaining Violation Data

First, specifying compliance requirements has certain limitations. In order to provide an all-round analysis, it should be possible to define properties of multiple cases (e.g. *At least half of the patients should be examined twice*). Now violations are always considered in isolation.

One of the implementation limitations is the necessity to specify temporal rules after running the analysis. Ideally, they should be specified in advance within the same Petri Net with Data as the rest of the requirements. In order to overcome this limitation, the way of defining the Petri Net with Data should also be changed.

Finally, the way rules have to be specified is complicated for users without knowledge of Petri Nets. This can be addressed by creating a more comprehensible, yet formal way to define rules, e.g. by answering a questionnaire. Another way is adapting some of the existing notations, or combining them.

8.2.2 Identifying the Relation Between Context Info and Violation

Violations concerning one activity might depend on some context information of a completely different activity. Yet, such a situation might be dealt with by adding a specific attribute, which would contain the context of the chosen activity into the event log. However, this would require hypothesizing from user, additional log refinement and would cause an increase in analysis complexity.

Another limitation is similar to the previous one: the user is forced to hypothesize about attributes, being relevant for violation and to choose which of them should be considered during the analysis of a particular problem. If the number of attributes is small, the user might be able to try different combinations. However, if there are many attributes, the user needs to be supported in choosing them. This brings another future work direction – guiding the user in selecting the attributes for comparison.

8.2.3 Visualizing Results for Business Users

Despite the fact that the presented results can be understood by users not having a technical background, it does need to be improved.

One possible way is changing the representation to be more visual. It might be possible to visualize a compliance rule using some simple process model notation (e.g. Fuzzy [2, 14] notation, or Disco[1] notation). This would allow the user to understand easily which parts of a process need to be examined more closely and improve general comprehension.

Another direction for future research is to enrich the obtained output with information about the consequences of every violation. For example, if an outgoing payment was performed by a

[1]http://fluxicon.com/disco/

wrong person, data of the amount of payment would help to understand whether the violation was serious and requires thorough investigation, or it would be sufficient to enforce rules more strictly.

Bibliography

[1] Decision tree learning. http://en.wikipedia.org/wiki/Decision_tree_learning. Accessed: 01-10-2013. 49

[2] Fuzzy Miner. http://www.processmining.org/online/fuzzyminer. Accessed: 07-11-2013. 76

[3] PwC Crime Survey 2011. http://crimesurvey.pwc.nl/. Accessed: 01-11-2013. 1

[4] SAP Help portal. http://help.sap.com/. Accessed: 10-01-2014.

[5] Wil M.P. van der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Springer, 2011. 2, 3, 7, 8, 16

[6] Wil M.P. van der Aalst, HT de Beer, and F. van Dongen, Boudewijn. Process mining and verification of properties: An approach based on temporal logic. 2005. 16

[7] Wil M.P. van der Aalst, Massimiliano de Leoni, and Arthur HM ter Hofstede. Process mining and visual analytics: Breathing life into business process models. *BPM Center Report BPM-11-15, BPMcenter. org*, 2011. 17

[8] Wil M.P. van der Aalst, Hajo A Reijers, and Minseok Song. Discovering social networks from event logs. *Computer Supported Cooperative Work (CSCW)*, 14(6):549–593, 2005. 8

[9] Wil van der Aalst, Arya Adriansyah, and Boudewijn van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012. 16

[10] Nouha Abid, Silvano Dal Zilio, and Didier Le Botlan. Real-time specification patterns and tools. In *Formal Methods for Industrial Critical Systems*, page 1–15. Springer, 2012. 16

[11] Arya Adriansyah, Boudewijn F. van Dongen, and Wil M.P. van der Aalst. Conformance checking using cost-based fitness analysis. In *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International*, page 55–64. IEEE, 2011. 16

[12] Johann Eder, Euthimios Panagos, and Michael Rabinovich. Time constraints in workflow systems. In *Advanced information systems engineering*, page 286–300. Springer, 1999. 16

[13] Ning Ge, Marc Pantel, and Xavier Crégut. Formal specification and verification of task time constraints for real-time systems. In *Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies*, page 143–157. Springer, 2012. 16

[14] Christian W Günther and Wil M.P. van der Aalst. Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *Business Process Management*, page 328–343. Springer, 2007. 76

[15] Kosmas Hatzidimitris. Using Visual Analytics for Conformance Checking and Compliance Rules. Master's thesis, Eindhoven University of Technology, 2013. 17

[16] Airlangga Adi Hermawan, M. de Leoni, and B Skoric. Context Analysis of Business Processes Based on Event Logs. Master's thesis, 2013. 17

[17] Massimiliano de Leoni and Wil M.P. van der Aalst. Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming. In *Business Process Management*, page 113–129. Springer, 2013. 16

[18] Massimiliano de Leoni and Wil M.P. van der Aalst. Data-aware process mining: discovering decisions in processes using alignments. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, page 1454–1461. ACM, 2013. 1, 16, 48, 49

[19] Massimiliano de Leoni, Wil M.P. van der Aalst, and Boudewijn F. van Dongen. Data-and Resource-Aware Conformance Checking of Business Processes. In *Business Information Systems*, page 48–59. Springer, 2012. 16, 23

[20] Massimiliano de Leoni, Fabrizio Maria Maggi, and Wil M.P. van der Aalst. Aligning event logs and declarative process models for conformance checking. In *Business Process Management*, page 82–97. Springer, 2012. 16

[21] Hongchen Li and Yun Yang. Dynamic checking of temporal constraints for concurrent workflows. *Electronic Commerce Research and Applications*, 4(2):124–142, 2005. 16

[22] J. Ross Quinlan. Improved use of continuous attributes in C4.5. *arXiv preprint cs/9603103*, 1996. 81

[23] John Ross Quinlan. *C4.5: programs for machine learning*, volume 1. Morgan kaufmann, 1993. 49, 81

[24] Elham Ramezani, Dirk Fahland, and Wil M.P. van der Aalst. Where did i misbehave? diagnostic information in compliance checking. In *Business Process Management*, page 262–278. Springer, 2012. 16, 21

[25] Elham Ramezani, Dirk Fahland, Boudewijn van Dongen, and Wil M.P. van der Aalst. Diagnostic information in temporal compliance checking. Technical report, Tech. rep. BPM Center Report BPM-12-17, BPMcenter. org, 2012. 15, 16, 22, 25, 52

[26] Anne Rozinat and Wil M.P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008. 16

[27] Patrícia Silveira, C Rodrguez, Aliaksandr Birukou, Fabio Casati, Florian Daniel, Vincenzo DAndrea, Claire Worledge, and Zouhair Taheri. Aiding compliance governance in service-based business processes. *Non-Functional Properties for Service-Oriented Systems: Future Directions (NFPSLA-BOOK-2011) Edition, IGI Global*, 2011. 16, 17, 75

[28] Michael Westergaard and Fabrizio Maria Maggi. Looking into the Future. In *On the Move to Meaningful Internet Systems: OTM 2012*, page 250–267. Springer, 2012. 16

[29] Xindong Wu, Chengqi Zhang, and Shichao Zhang. Efficient mining of both positive and negative association rules. *ACM Transactions on Information Systems (TOIS)*, 22(3):381–405, 2004. 15, 16

Appendix A

Description of C4.5 Algorithm

In this appendix we show description of C4.5 decision tree building algorithm as it is given in [22].

C4.5 uses a divide-and-conquer approach to growing decision trees. Only a brief description of the method is given here; see [23] for a more complete reference. As described in [22], the following algorithm generates a decision tree from a set D of cases:

- If D satisfies a stopping criterion, the tree for D is a leaf associated with the most frequent class in D. One reason for stopping is that D contains only cases of this class, but other criteria can also be formulated.
- Some test T with mutually exclusive outcomes $T_1, T_2, \ldots, T_k$ is used to partition D into subsets $D_1, D2, \ldots, D_k$, where D_i contains those cases that have outcome T_i. The tree for D has test T as its root with one subtree for each outcome T_i that is constructed by applying the same procedure recursively to the cases in D_i.

Provided that there are no cases with identical attribute values that belong to different classes, any test T that produces a non-trivial partition of D will eventually lead to single-class subsets as above. However, in the expectation that smaller trees are preferable (being easier to understand and often more accurate predictors), a family of possible tests is examined and one of them chosen to maximize the value of some *splitting criterion.* Here we show only the criterion for discrete values:
$A =?$ for a discrete attribute A, with one outcome for each value of A. Please refer to the original article for description of the criterion for a continuous attribute.

The default splitting criterion used by C4.5 is gain ratio, an information-based measure that takes into account different numbers (and different probabilities) of test outcomes. Let C denote the number of classes and $p(D, j)$ the proportion of cases in D that belong to the jth class. The residual uncertainty about the class to which a case in D belongs can be expressed as

$$Info(D) = -\sum_{j=1}^{C} p(D,j) \times log_2(p(D,j))$$

and the corresponding information gained by a test T with k outcomes as

$$Gain(D,T) = Info(D) - \sum_{i=1}^{k} \frac{|D_i|}{D} \times Info(D_i)$$

The information gained by a test is strongly affected by the number of outcomes and is maximal when there is one case in each subset D_i. On the other hand, the potential information obtained by partitioning a set of cases is based on knowing the subset D_i into which a case falls; this *split information*

$$Split(D,T) = -\sum_{i=1}^{k} \frac{|D_i|}{D} \times log_2(\frac{|D_i|}{D})$$

tends to increase with the number of outcomes of a test. The gain ratio criterion assesses the desirability of a test as the ratio of its information gain to its split information. The gain ratio of every possible test is determined and, among those with at least average gain, the split with maximum gain ratio is selected.

In some situations, every possible test splits D into subsets that have the same class distribution. All tests then have zero gain, and C4.5 uses this as an additional stopping criterion.

Appendix B

Compliance Framework Evaluation Questionnaire

Summarize you professional experience	
Area of expertise	**Years of experience**

Scenario 1		
Run the analysis on the data in folder ‘Scenario1’. Leave the default mapping for events and attributes. Answer questions 1 - 3		
	Question	**Answer**
1	List the violated activities	
2	Which **MATERIAL** is likely to have connection with the violations?	
3	What are the common attributes’ values between violations of the 2 checked rules?	

Run the problem insight plugin on the problem ‘*Activity OutgoingPayment is performed with incorrect* ***PO_AMOUNT****, when* ***MATERIAL*** *= material607*’. In the tab ‘Same attribute value’ uncheck all attributes, except **USER_ID.** Answer the question 4.		
4	List the users for which events occur with incorrect **PO_AMOUNT**	
Scenario 2		
Run the analysis on the data in folder ‘Scenario2’. Leave the default mapping for events and attributes. Answer the questions 5 – 6		
5	List the **MATERIAL**s involved in the violation	
6	In your opinion, which attribute has stronger relation to the problem: **PO_AMOUNT**, or **ENTRY_AMOUNT**?	
Run the problem insight on ‘*Activity GoodsReceipt is performed with incorrect* ***USER_ID***’. Leave the default settings for configuration. Answer the question 7.		
7	What are the users involved into the violations?	
Scenario 3		
Run the analysis on the data in folder ‘Scenario3’. Leave the default mapping for events and attributes. Set **MATERIAL** importance to 0. Answer the question 8.		
8	Set **MATERIAL** importance to 0, so that it wouldn’t flood the report. Which users are responsible for the violations?	
Run the ‘Get Problem Insight’ plugin on the problem ‘*Activity OutgoungPayment is performed with incorrect* ***USER_ID***’. Among the attributes, leave only **ENTRY_AMOUNT** and **USER_ID**, and set ‘Considering Time’ to ‘Disabled’. Leave the rest as default. Answer the question 9		
9	What is the maximal **ENTRY_AMOUNT** for which the problem occurs?	

Run the ‘Get Problem Insight’ plugin on the problem ‘*Activity OutgoungPayment is performed with incorrect **USER_ID***’. Among the attributes, leave only **PO_AMOUNT** and **USER_ID**, and set ‘Considering Time’ to ‘Disabled’. Leave the rest as default. Answer the question 10.		
10	Based on the confidence, which amount, **ENTRY_AMOUNT** (from the previous question) or **PO_AMOUNT** has stronger relation to the problem?	

General Observations and comments: