SN Computer Science (2024) 5:706
https://doi.org/10.1007/s42979-024-03008-8

ORIGINAL RESEARCH

Analyzing Data Streams from Cyber-Physical-Systems: A Case Study

Harry H. Beyel[1] · Omar Makke[2] · Mahsa Pourbafrani[1] · Oleg Gusikhin[2] · Wil M. P. van der Aalst[1]

Received: 12 December 2023 / Accepted: 22 May 2024

Abstract
We show that conducting a process-mining-centric analysis concerning cyber-physical systems provides insights into usage behavior. To show that, we perform our analysis on connected-vehicle data. We transform connected-vehicle data into an event log. We analyze the resulting event log using various process-mining techniques. In particular, we apply basic statistical analysis as well as process-discovery and conformance-checking techniques to receive a well-representative process model. We apply various process-enhancement techniques to get deeper insights. Finally, we capture a multi-perspective view using a state-based approach. We show deviations between a de-jure model and our picked process model, leading to better knowledge concerning real user behavior. We observed that the predefined escalation of warning states does not happen. Additionally, we verified system requirements. Furthermore, we show that the reasons for drivers' behavior are not related to system issues. Applying process-mining techniques to data concerning cyber-physical systems provides valuable insights into their functionality in a real-world setting. By utilizing process-mining techniques, we can extract insights to a human-understandable level and provide a well-studied access point.

Keywords Process mining · Connected vehicles · Internet of Things · Cyber-physical systems

Introduction

Cyber-Physical Systems (CPSs) [1] and Internet of Things (IoT) [2] are omnipresent and are integrated into our daily lives. While CPSs are about integrating physical components with computing and controlling systems for autonomous operation in real time, IoT focuses on the interconnection of a wide range of devices and objects for data collection and communication. These systems are implemented in various domains, from household items to connected vehicles. By using these devices, an enormous volume of data is generated. When we analyze the data to capture the overall systems' behavior and the underlying human interactions, we face multiple challenges. One challenge is that the data from IoT devices are often captured by sensors, which transmit their values, for instance, an acceleration or the heading of an object. We refer to such data as low-level data. By analyzing the data, we want to understand the higher-level interactions between humans and these devices. As a result, there is a gap between the recorded data and our goal.

Process-mining techniques analyze data to support the understanding of the underlying behavior. Process mining can be divided into three areas: process discovery, conformance checking, and process enhancement [3]. Process-discovery techniques aim to automatically discover a comprehensible process model that represents the underlying behavior in the data by focusing on the control-flow perspective. Conformance-checking techniques quantify how well a process model represents the behavior in the recorded data. Process-enhancement techniques are used to generate deeper insights based on a well-representative process model, such as measuring the time spent in states or reasons for decisions. To apply process-mining techniques, data must be in the form of an *event log*. An example event log can be

✉ Harry H. Beyel
beyel@pads.rwth-aachen.de

Omar Makke
omakke@ford.com

Mahsa Pourbafrani
mahsa.bafrani@pads.rwth-aachen.de

Oleg Gusikhin
ogusikhi@ford.com

Wil M. P. van der Aalst
wvdaalst@pads.rwth-aachen.de

[1] Chair of Process and Data Science, RWTH Aachen University, Aachen, Germany

[2] Global Data Insight & Analytics, Ford, Dearborn, USA

Published online: 10 July 2024

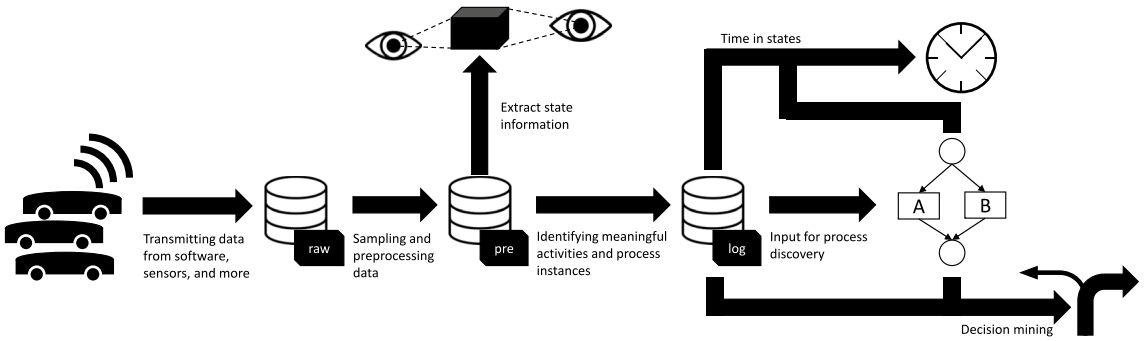


Fig. 1 Overview of our framework for transforming vehicle data and applying process-mining techniques

extracted from an airport system. Activities are "checking the luggage", "scanning the passport", etc., and a customer serves as an identifier, trackable in the data. These activities are sequences based on their execution time, i.e., timestamp. However, data generated by IoT devices are not in the format of an event log. Data produced by IoT devices may only consist of low-level measurements and timestamps that reveal when the record happened. An example of such low-level data is transmitted by a temperature sensor attached to a connected vehicle. Such data must be transformed into an event log where a measurement is associated with some identifier that can be tracked throughout the system.

As denoted in [4] and in our previous example, using IoT data for process mining is a challenging task. In this work, we provide a framework of how data transmitted by test vehicles can be transformed into an event log and which insights process mining can provide. We focus on a part of a hands-free driving system that is embedded in connected test vehicles. The system enables the vehicle to be driven hands-free and without human intervention in defined zones as long as defined conditions are not violated. Vehicles equipped with this feature require connectivity and can receive over-the-air updates to improve the feature's behavior based on its usage. On connected test vehicles equipped with a hands-free driving system, a data stream is recorded at a one-second rate and transmitted to the cloud using the vehicles' modem. Features of the collected data include an anonymized vehicle identifier and states and warnings of assistance systems. By utilizing the data for process mining, we can get insights into the usage of the system and the system's behavior. These insights can guide future updates of the hands-free driving system and help to spot the needs of users and flaws in the system.

As stated, IoT data are challenging for process-mining approaches. In addition to these challenges, data transmitted by vehicles adds more challenges when we apply process-mining techniques. First, the data collection is susceptible to *data quality* issues and noise. At any instant in time, thousands of vehicles send data under different conditions, such as software versions, sensors, ages, and connectivity conditions. An example of inaccurate information is GPS signals contradicting vehicle speed or temperature measurements taken while a vehicle is parked under the sun. Second, *privacy* and *security* have to be ensured. The collected data may contain information that could be used to identify individuals directly or indirectly. Preserving the privacy and security of individuals' data is vital. Third, the *contextualization* of data is not trivial. Without additional context, the recorded data may not be well interpretative. For example, when the speed of a car is low, potential reasons can be road conditions or a traffic jam. Without this contextualization, it is hard to understand the behavior of drivers. Our work is able to resolve some of these challenges.

In this work, we focus on analyzing real-life test vehicle data from Ford by transforming the data and utilizing various process-mining techniques. An overview of our approach is depicted in Fig. 1.

As shown, we take a sample of the recorded test vehicle data. This sample of connected-vehicle data is transformed into an event log, suitable for process mining. We perform an analysis of the event log. By applying process-mining techniques, we receive a well-representative process model that we use to generate further insights. Finally, we capture a multi-perspective view of the process.

In the remainder of this paper, we first show and discuss related work in the Sect. "Related work". In the Sect. "Preliminaries", we introduce basic concepts. In the Sect. "Vehicle Data", we provide insights into how we transformed vehicles' data into an event log. In the Sect. "Statistics", we analyze the received event log using a basic statistical analysis. In the Sect. "Process Analysis" , we perform a process analysis using process-mining techniques. We capture a multi-perspective view of the process in the Sect."Multi-perspective Analysis" section. We summarize and discuss our work in the Sect. "Conclusion" and describe future work.

Related Work

In this section, we present and comment on related work. In [4], challenges and opportunities for IoT by using IoT data for process mining are presented. IoT has many challenges, such as sensor placement and data connectivity, security, and privacy. Bridging the gap between low-level sensor data and event logs leads to more challenges. An example of a challenge is identifying activities and processes associated with the recorded data. In this work, we are facing such challenges. In the following, we first present related work concerning the general intersection of BPM and IoT. Subsequently, we focus on the intersection of BPM and IoT by focusing on a vehicle-related setting.

An example of why tackling and overcoming these challenges is rewarding is presented in [5]. In [5], a case study on integrating Business Process Management (BPM) and IoT in Australia's meat and livestock industry is presented. This study does not describe how challenges were overcome, but the benefits of combining BPM with IoT are presented. An example is that farmers can confidently plan daily activities through the embedded system's support. Based on the results of our study, we demonstrate that original equipment manufacturers such as Ford can benefit by combining BPM with IoT. The understanding of the usage of the hands-free driving feature improves, and the underlying system can be checked. The system can be improved, validated, and adjusted to users' needs based on the overall better insights. In [6], van Eck et al. segment sensor data in smaller time windows and compute relevant features for each segment, for example, the average value of an attribute. These relevant features are used to cluster the segments. Each cluster is assigned an activity by using domain knowledge. In our work, we use sensor recordings to create additional information for each event, for instance, if the vehicle is facing the sun. Also, we use domain knowledge. However, we do not rely on time windows. Instead, we focus on the change of attribute values. In [7], a general framework to discover process models from sensor data is presented. The work utilizes location sensor data. Events are grouped into activity instances through correlation. Afterward, activities are discovered. Then, events are abstracted based on the groups and the discovered activities. Finally, processes can be discovered. Our study does not deal with location data, but we also have to define activities and process instances. In [8], data from user interactions with software are collected and analyzed. Mainly, process-discovery techniques are used for the analysis. Based on the results of the analysis, improvements for the software are defined. We also aim to improve the understanding of user behavior to implement improvements. Additionally, we want to find out the reasons behind the system's or drivers' decisions. Therefore, we do not rely only on process discovery; we also perform process enhancement techniques.

There also exist works in the intersection of vehicles and process mining. In [9], a Petri net is discovered to track the occupancy of a parking space. In addition, the discovered model can be adjusted during deployment, for instance, if a sensor is broken. Another example is the work presented in [10], for which this work is an extension. In [10], vehicle data are transformed into event data. In addition, a de-jure model is created by consulting domain experts. Besides, de-facto models are discovered by applying dedicated process-discovery techniques to the received event data. The most promising process model is picked by considering conformance-checking results and opinions of domain experts. The picked de-facto model is compared against the de-jure model by considering the conformance-checking result and their behavior. In this work, we also discover de-facto models but on a different data set. Moreover, we apply dedicated process-enhancement techniques. Also, we capture a multi-perspective view of the data. In [11], process enhancement techniques are in focus to gain more insights. A comparison between vehicle models concerning time spent in states is conducted. Also, the mentioned work discovers statistical differences in the execution frequency of activities between vehicle models. In addition, decision mining is applied to a process model to discover reasons for decisions, for example, why the hands-free-driving feature was turned off instead of solving an issue. In our work, we perform different tasks of process-enhancement techniques since we have different data. Also, we capture a multi-perspective view of the process, which was not performed in the mentioned work.

Preliminaries

In this section, we formally introduce concepts that are the basics of the techniques we propose later. Given a set X, a sequence $\sigma \in X^*$ assigns an enumeration to elements of the set. We denote this with $\sigma = \langle \sigma_1, ...\sigma_n \rangle$. If a sequence is a subsequence, we denote this with $\sqsubseteq$. In the remainder, we refer with σ_i to the sequence's i-th element.

To apply process-mining techniques, we need an event log. An event log consists of at least three mandatory attributes: *case identifier*, *activty name*, and *timestamp*. We use $\mathcal{U}_{case}$ as the universe of case identifiers, $\mathcal{U}_{act}$ as the universe of activity names, and $\mathcal{U}_{time}$ as the universe of timestamps. The following defines an event log.

Definition 1 (*Event log*) $\mathcal{U}_{ev}$ is the universe of events. $e \in \mathcal{U}_{ev}$ is an event, $\pi_{act}(e) \in \mathcal{U}_{act}$ is the activity of e, $\pi_{case}(e) \in \mathcal{U}_{case}$

Table 1 Example event log

Event ID	Case	Activity	Timestamp
e_1	1337	a	2023-01-21
e_2	1337	b	2023-02-15
e_3	1337	c	2023-05-05
e_4	1338	d	2023-06-01
e_5	1338	e	2023-06-19
e_6	1338	c	2023-07-20

is the case of e, and $\pi_{time}(e) \in \mathcal{U}_{time}$ is the timestamp of e. An event log $L \subseteq \mathcal{U}_{ev}$ is a set of events. For simplicity, we assume that other, here non-defined, functions can be applied to an event, resulting in more attributes.

An example event log is displayed in Table 1. For e_1, $\pi_{case}(e_1) = 1337$, $\pi_{act}(e_1) = a$, and $\pi_{time}(e_1) = 2023\text{-}01\text{-}21$.

In the remainder of this work, we use Petri nets as process model representations. They are introduced in [3, 12]. We also use τ-transitions, respectively, silent transitions. In Petri nets, silent transitions provide a flexible and expressive mechanism for modeling various aspects of process behavior. They enable the representation of internal actions, improving Petri nets' modeling capabilities. They allow the synchronization of different parts of the Petri net and can be used to control the execution of other transitions by requiring that certain conditions or constraints be met before proceeding. Silent transitions can act as synchronization points, waiting for specific conditions to be met before allowing subsequent transitions to fire. In our figures, we depict them as black transitions. We use conformance-checking techniques to quantify a process model's quality by measuring fitness, precision, simplicity, and generalization scores. [13] provides an introduction to these measurements.

Vehicle Data

In this section, we focus on connected-vehicle data from Ford's test vehicles. First, we describe the data and how they were gathered. Second, we describe the preprocessing of the data. Finally, we describe the transformation of vehicle data into an event log.

Data Description and Gathering

We receive data transmitted by test vehicles in the United States of America. The data of vehicles are recorded on US highways at a sample rate of roughly one second. The data contains information related to cars' software and sensors. The following briefly explains the recorded data's most important features.

- *Vehicle*: Vehicle to which a datum belongs.
- *Journey*: A journey is created as soon as a recording of a vehicle takes place. Moreover, journeys are generated if highways are switched. In addition, journeys are generated if a vehicle drives in a rest area or at a petrol station. Consequently, traveling can involve multiple journeys. A journey is linked to a vehicle.
- *State*: States of traffic jam assist. We consider four states: hands-free driving (HF) and "State 1", "State 2", and "State 3".
- *Warning*: Different warnings are stored in this feature. We consider the following: "No Warning", "Warning 1", "Warning 2", and "Warning 3".
- *Longitude*: Part of the GPS coordinate of the vehicle.
- *Latitude*: Part of the GPS coordinate of the vehicle.
- *Heading*: Showing the direction in which the vehicle is heading.
- *Wiper*: State of the front wiper of the vehicle.
- *Model*: Model of the vehicle.
- *Year*: Production year of the vehicle.
- *Plant*: Plant in which the vehicle was produced.
- *Timestamp*: Timestamp for each recording.

An example of such data is shown in Table 2. The size of the collected data is large. Unfortunately, it is difficult for most process-mining tools to process data at such a large scale. Systems like Celonis can handle billions of events, but in this work, we are using open-source software running locally. To overcome this issue, we sample data. Journeys are randomly sampled from the collected data, and only journeys with useful information are considered to be part of the sampling pool, thus minimizing the amount of erroneous data. Our sampled data contains over 8,635,640 instances, 10,733 journeys, and records from over 500 vehicles. The second step is filtering the data. Due to the large amount of data available, we found the results more revealing if we only keep journeys with hands-free engagements. Also, through several iterations, we identified some erroneous data, such as overlapping journeys. Therefore, we created a script based on domain knowledge that checks the data for flaws.

In the following, we formally introduce vehicle data. $\mathcal{U}_{journey}$ is the universe of journey identifiers, S is the set of states, and W is the set of warnings.

Definition 2 (*Vehicle data*) $\mathcal{U}_{vd}$ is the universe of vehicle data. $d \in \mathcal{U}_{vd}$ is a vehicle data instance, $\pi_{journey}(d) \in \mathcal{U}_{journey}$ is the journey of d, $\pi_{state}(d) \in S$ is the state of d, $\pi_{warning}(d) \in W$ is the warning of d, and $\pi_{time}(d) \in \mathcal{U}_{time}$ is the timestamp of d.

Table 2 Example recording of vehicle data

RowID	Vehicle	Journey	State	Warning	Longitude	Latitude	Heading	Wiper	Model	Year	Plant	Timestamp
1	ABC	1	HF	No warning	60.01	30.02	65.21	Off	A	2023	Z	2023-01-01 13:37:37
2	ABC	1	HF	No warning	60.01	30.03	65.22	Off	A	2023	Z	2023-01-01 13:37:38
3	ABC	1	HF	Warning 1	60.02	30.03	65.23	Off	A	2023	Z	2023-01-01 13:37:39
4	ABC	1	HF	Warning 1	60.02	30.02	65.21	Off	A	2023	Z	2023-01-01 13:37:40
5	ABC	1	HF	No warning	60.02	30.03	65.21	Off	A	2023	Z	2023-01-01 13:37:41
6	ABC	1	State 1	No warning	60.01	30.04	65.21	Off	A	2023	Z	2023-01-01 13:37:42
7	ABC	1	State 1	No warning	60.01	30.04	65.24	Off	A	2023	Z	2023-01-01 13:37:43
8	ABC	1	HF	No warning	60.02	30.04	65.22	Off	A	2023	Z	2023-01-01 13:37:44
9	ABC	1	HF	No warning	60.02	30.05	65.24	Off	A	2023	Z	2023-01-01 13:37:45
10	ABC	1	State 1	No warning	60.03	30.05	65.25	Off	A	2023	Z	2023-01-01 13:37:46
11	ABC	1	State 1	No warning	60.03	30.05	65.26	Off	A	2023	Z	2023-01-01 13:37:47
12	ABC	2	HF	No warning	61.05	29.92	15.29	Off	A	2023	Z	2023-01-02 20:08:02
13	ABC	2	HF	No warning	61.05	29.93	15.29	Off	A	2023	Z	2023-01-02 20:08:03
14	ABC	2	HF	No warning	61.05	29.94	15.31	Off	A	2023	Z	2023-01-02 20:08:04
15	ABC	2	State 2	No warning	61.05	29.95	15.32	Off	A	2023	Z	2023-01-02 20:08:05

The column names are related to the formerly mentioned features. "RowID" is used to identify instances

Preprocessing

We preprocess the data before we transform the vehicle data into an event log. Our preprocessing consists of multiple steps. First, hands-free driving should occur at least once in each journey since we analyze the corresponding feature. If this is not the case, we remove the journey. Second, we check if a vehicle's journeys overlap due to potential logging issues. If so, we remove the journey from the data. Third, based on the system's nature, two journeys may be created, even though not much time has passed between the two journeys. Therefore, we merge trips of a vehicle if they are less than 5 s apart. Besides, we added information on the sun's position relative to the vehicle for each entry. The sun is either left, right, in the front, or in the back of the vehicle. To compute this information, we utilized the GPS information of a record, i.e., latitude, longitude, and the direction the vehicle faces (azimuth), combined with the information on when the record happened. We use the method presented in [14]. This method is implemented in the Python library *pvlib* [15]. After preprocessing the vehicle data, we ended up with 1,342,825 instances and 1055 journeys from over 400 vehicles.

Transforming Vehicle Data into an Event Log

After preprocessing the data, we have to transform the data into an event log to apply process-mining techniques. For the transformation, journeys play a vital role. As introduced, they provide an identifier that enables the linkage of vehicle data and provides a first separation of the data stream. However, they are too coarse for our analysis scope. Therefore, we divide them into smaller units, which we call *runs*. Each run represents a driver's engagement with the feature, i.e., to analyze the feature, a run provides a perfect scope, which we use later as an event log's case. Based on our scope, some instances of our data are not of interest and are not part of a run. Instances that are part of a run are assigned activities. After filtering the created runs, we transform the remaining vehicle data and create an event log suitable for process mining. In order to transform the set of vehicle data, the presented methods have to be applied on all journeys and, consequently, on all runs.

From Journeys to Runs

In order to explain the transformation process, we need to extract data from our set of vehicle data that are associated with one journey. Our transformation approach is based on sequences, so we transform the data associated with a journey into a sequence in which the data are sorted by their timestamp—from the earliest to the latest.

Definition 3 (*Journey sequence*) Let $D \subseteq \mathcal{U}_{vd}$ be a set of vehicle data. $D_{journey} = \{d \in D \mid \pi_{journey}(d) = journey\}$ is a subset of D that only contains instances that are associated with one journey. $\sigma_{journey} = \langle d_1, \ldots, d_n \rangle$, $d_1 \in D_{journey}, \ldots, d_n \in D_{journey}$, is the sequential representation of $D_{journey}$ such that $\pi_{time}(d_1) < \cdots < \pi_{time}(d_n)$.

Given our vehicle data depicted in Table 2, $D_1 = \{d_1, \ldots, d_{11}\}$ and $D_2 = \{d_{12}, \ldots, d_{15}\}$.

We want to capture the beginning of the hands-free driving mode until its end. To do so, we divide the sequence of

a journey into runs. Each run starts with the beginning of hands-free driving and ends with the first state that is not hands-free.

Definition 4 (*Run*) Let $D \subseteq \mathcal{U}_{vd}$ be a set of vehicle data, and $D_{journey}$ and $\sigma_{journey} = \langle d_1, \dots, d_n \rangle$ be defined as before. $\langle d_i, \dots, d_j \rangle \sqsubseteq \sigma_{journey}$, with $j > i$, is a run if $\pi_{state}(d_{i-1}) \neq \text{HF}$ or $i = 1$. Furthermore, $\pi_{state}(d_i) = \text{HF}, \dots, \pi_{state}(d_{j-1}) = \text{HF}$ and $\pi_{state}(d_j) \neq \text{HF}$. We refer to such a sequence as σ_{run}, with $\pi_{run}(d_i) = run, \dots, \pi_{run}(d_j) = run$. The data of one run is stored in D_{run}, i.e., $D_{run} = \{d \in D \mid \pi_{run}(d) = run\}$.

One run that we observe for D_1 is $\langle d_2, \dots, d_6 \rangle$. Another run is $\langle d_8, d_9, d_{10} \rangle$.

Assigning Activities

In the next step, we assign each vehicle datum that is associated with a run an activity. While the first instance of a run is assigned "System started" and the last the state of the vehicle, we focus on warning changes for the rest of the data in a run. An absence of information is denoted with $\perp$. The following presents our activity assignment.

Definition 5 (*Creating activities*) Let $D \subseteq \mathcal{U}_{vd}$ be a set of vehicle data, and D_{run} and $\sigma_{run} = \langle d_1, \dots, d_n \rangle$ be as introduced before. We assign each d_i, $i \in \{1, \dots, n\}$, an activity as follows:

$$\pi_{act}(d_i) = \begin{cases} \text{System started}, & \text{if } i = 1 \\ \pi_{state}(d_i), & \text{if } i = n \\ \pi_{warning}(d_i), & \text{if } \pi_{warning}(d_i) \neq \text{No Warning} \\ & \text{and } \pi_{warning}(d_i) \neq \pi_{warning}(d_{i-1}) \\ \text{Warning solved}, & \text{if } \pi_{warning}(d_i) = \text{No Warning} \\ & \text{and } \pi_{warning}(d_{i-1}) \neq \text{No Warning} \\ \perp, & \text{else} \end{cases}$$

Concerning the run $\langle d_2, \dots, d_6 \rangle$, d_2 gets assigned "System started", d_3 gets assigned "Warning 1", d_5 gets assigned "Warning solved", and d_6 gets assigned "State 1". The others are assigned $\perp$ as the activity.

Filtering Runs

So far, instances that are associated with a run are assigned activities. However, some instances are assigned $\perp$ as activity. Instances assigned with this activity do not provide value for our analysis. Therefore, we filter them out.

Table 3 Event log based on transformed example recording of vehicle data

RowID	CaseID	Activity	Vehicle ID	..	Timestamp
1	001	System started	ABC	..	2023-01-01 13:37:37
3	001	Warning 1	ABC	..	2023-01-01 13:37:39
5	001	Warning solved	ABC	..	2023-01-01 13:37:41
6	001	State 1	ABC	..	2023-01-01 13:37:42
8	002	System started	ABC	..	2023-01-01 13:37:44
10	002	State 1	ABC	..	2023-01-01 13:37:46
12	003	System started	ABC	..	2023-01-02 20:08:02
15	003	State 2	ABC	..	2023-01-02 20:08:05

Definition 6 (*Filtering runs*) Let $D \subseteq \mathcal{U}_{vd}$ be a set of vehicle data, and D_{run} and $\sigma_{run} = \langle d_1, \dots, d_n \rangle$ be as introduced before. $D^{filer}_{run} = \{d \in D_{run} \mid \pi_{act}(d) \neq \perp\}$ is the data associated with a run that has not $\perp$ assigned as activity. σ^{filter}_{run} the sequential representation of D^{filter}_{run} sorted from earliest to latest.

Based on the run $\langle d_2, \dots, d_6 \rangle$, the filtered run is $\langle d_2, d_3, d_5, d_6 \rangle$. The run $\langle d_8, d_9, d_{10} \rangle$ is reduced to $\langle d_8, d_{10} \rangle$.

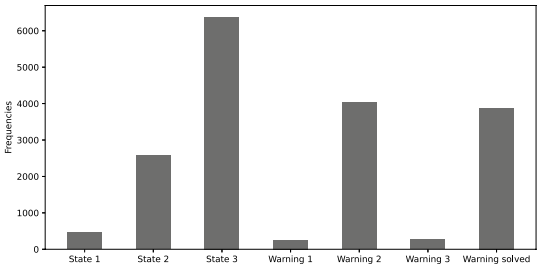
Transformation

In the last step, we transform the vehicle data into an event log. To do so, we focus on vehicle data of a filtered run.

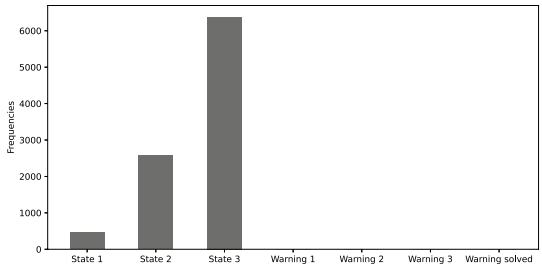
Definition 7 (*Transformation*) Let $D \subseteq \mathcal{U}_{vd}$ be a set of vehicle data, $L \subseteq \mathcal{U}_{ev}$ be an event log, and D^{filter}_{run} and $\sigma^{filter}_{run} = \langle d_1, \dots, d_n \rangle$ be as previously introduced. There exists a function $trans : D^{filter}_{run} \rightarrow L$ that maps each element of D^{filter}_{run}, $d \in D^{filter}_{run}$, to an event, $e \in L$, such that the following holds:

- $trans(\pi_{run}(d)) = \pi_{case}(e)$
- $trans(\pi_{act}(d)) = \pi_{act}(e)$
- $trans(\pi_{time}(d)) = \pi_{time}(e)$

Concerning our previously introduced example vehicle data, the transformed event log is shown in Table 3.



(a) Frequency distribution of all activities across the cases.



(b) Frequency distribution of endactivities across the cases.

Fig. 2 Distributions of activities in the transformed event log

Statistics

In this section, we analyze the event log that we transformed from the vehicle data. First, we filter the event log. Second, we analyze the frequency of activities. Third, we take a closer look at the variants of the filtered event log. Finally, we provide a short overview of time-related statistics.

Filtering

The event log we received by applying our previous steps on vehicle data still contains noise, which we need to remove. Our filtering is based on domain knowledge and consists of two parts. First, if an event with activity "Warning 3" happens, only an event with activity "State 1", "State 2", or "State 3" can take place next in a case. Second, a case has to end with an event with activity "State 1", "State 2", or "State 3". Non-compliant behavior occurs due to logging issues or actions taking place between the transmitting period, thus not appearing in the data. By only using this limited set of rules, we ensure that we observe the system's and drivers' real behavior. In this process, we remove $\approx 2.6\%$ of traces, leading to an event log consisting of 9413 traces and 27,266 events. Removing only this fraction of traces shows that the transmitted data quality is good concerning the control flow.

Analysis of Activities

After filtering the transformed vehicle data, we take a look at the various activity frequencies'. In this process, we focus on the frequency of end activities and the overall frequency. Due to our transformation, we know the start activity is "System started". An overview of the distributions is depicted in Fig. 2.

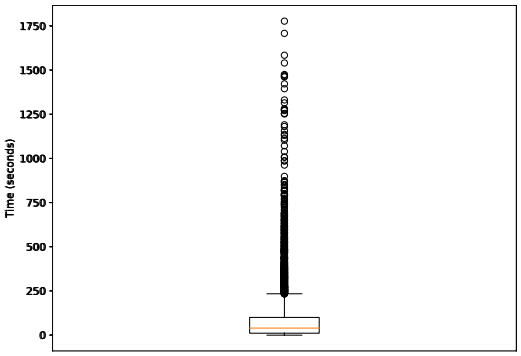
In Fig. 2a, the distributions of activities across all cases contained in the transformed event log are shown. "System started" is not shown since it appears in each case once, leading to a frequency of 9413. When considering the frequencies of the different warnings, it becomes visible that "Warning 2" happens way more often than "Warning 1" and "Warning 3" combined. Since the different warnings are related to an escalation, it becomes clear that there may be issues. When focusing on the frequency relationship between "Warning solved" and the different warnings, we see that not all warnings are solved. As a result, there are other strategies that we will explore in the remainder of this paper. In Fig. 2b, the distribution of end activities is shown. Important to note is that the frequency of "State 1", "State 2", and "State 3" is the same as depicted in Fig. 2a since these activities serve as end activities. The figure shows that most of the time "State 3" turns the system off, followed by "State 2". Only $\approx 5\%$ of the time, "State 1" is executed.

This analysis gave us a first overview of how the system operates when considering the transmitted data. We found that "Warning 2" is the most executed warning (with an execution probability of $\approx 43\%$ per case). Additionally, we denoted that not all warnings are solved, indicating that there are other strategies to get out of the warning state.

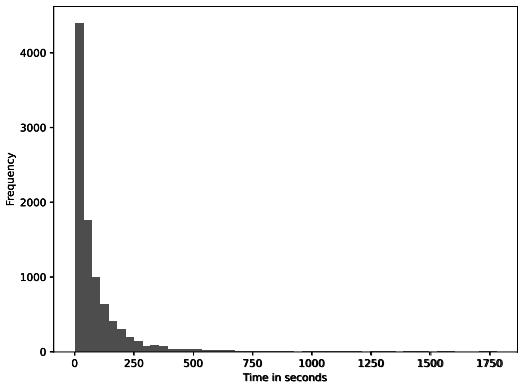
Analysis of Variants

In the following, we take a look at the variants of the event log. For this analysis, we focus on the variants obtained by focusing on the order of activities. In total, 202 variants are contained in the received event log. In the following, we list the five most prominent variants and their frequency, absolute and relative.

1. ⟨System started, State 3⟩: 55.43% (5,218 traces)
2. ⟨System started, State 2⟩: 23.38% (2,201 traces)
3. ⟨System started, Warning 2, Warning solved, State 3⟩: 4.80% (452 traces)
4. ⟨System started, State 1⟩: 3.63% (342 traces)



(a) Boxplot concerning the time spent in hands-free driving.



(b) Histogram about the spent time.

Fig. 3 Visualizations concerning the time spent in hands-free driving

5. ⟨System started, Warning 2, Warning solved, State 2⟩: 2.01% (189 traces).

The cases related to the listed variants cover 89.25% of cases of the event log. Therefore, they are worth taking a closer look. Approximately 82% of the time, the hands-free driving feature is started and turned off without a warning state. Hence, in 18% of the cases, warnings have to happen at least once. The two most prominent variants of these cases are presented in our enumeration. In each, "Warning 2" happens and is resolved. Later, the system is turned off through "State 2" or "State 3". Besides, we denote that "Warning 1" is not part of the variants, explaining its previously shown low frequency.

Time-Related Statistics

In the next step, we focus on time-related statistics. In particular, we investigate how much time is spent in the hands-free driving mode. To do so, we measure the time passed in the filtered event log cases. In addition, we filter out measured times if they are greater than 2000s (≈ 33 minutes) to show a comprehensive overview. With this filtering, we ensure to capture more than 99.3% of measured times, thus not removing the actual behavior. An overview of the measured times is provided in Fig. 3.

As shown in Fig. 3, most cases have a throughput time of roughly four to five minutes. The first quartile is ≈ 11 s, the median is ≈ 40 s, the mean is ≈ 82 s, and the third quartile is ≈ 100 s. It is revealed that most cases, i.e., runs of the system, cover only a short period. To understand what happens in this short amount of time, we need to analyze the process.

Process Analysis

In this section, we perform a process-oriented analysis. Our analysis consists of three parts. First, we decide on a well-representative process model by considering the available data and knowledge of domain experts. Second, by utilizing conformance checking, we get a closer look at the behavior of the hands-free driving feature. Third, we analyze the performance utilizing the picked process model. Fourth, we discover reasons for decisions by using decision mining.

Finding a Suitable Process Model

This subsection shows our process of finding a representative process model. We use multiple approaches. We start by constructing a de-jure model based on the system documentation. Subsequently, we discover de-facto process models based on our transformed event log. Finally, we evaluate the models using domain experts' opinions and metrics.

Constructing the De-jure Model

To implement the feature of hands-free driving, a behavioral model in the form of a flow chart was designed by humans. In our work, we focus on a subset of the system. This subset includes a small selection of warnings and conflict resolution strategies. We focus on conflicts related to the absence of focus of a driver and three warning types. The absence of focus is measured by checking if the driver still looks at the road. The textual description of the model is as follows. Assume the starting state is hands-free driving without interruption. If the eyes are off the road, a first warning goes

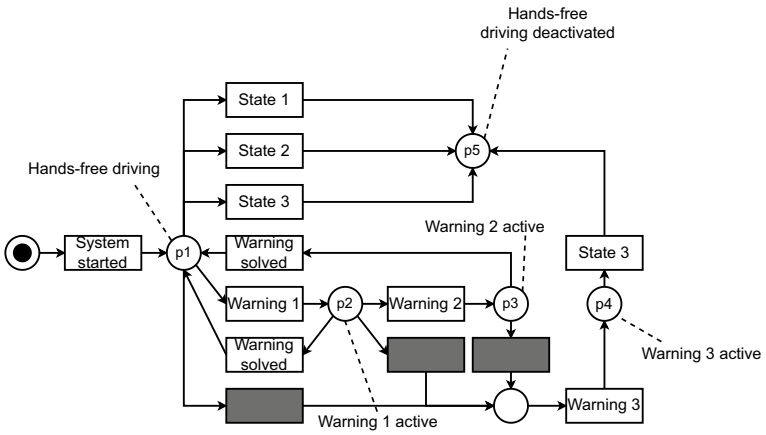


Fig. 4 Petri net showing the behavior of the hands-free driving system based on human design. After executing "System started", the vehicle is in the state of hands-free driving

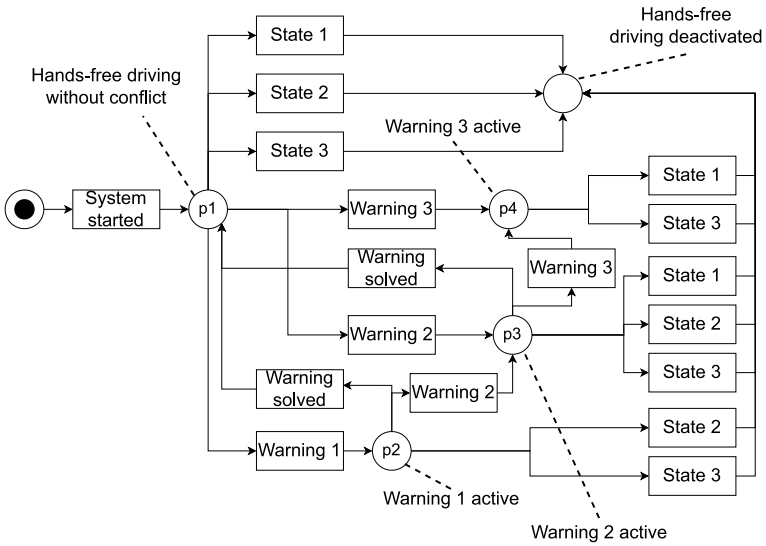


Fig. 5 Petri net discovered by applying region-based mining

off. The driver can resolve this conflict by looking back on the road. A second warning appears if the conflict is not resolved, meaning eyes are still not looking on the road. Again, the situation can be solved by looking back on the road. If that is not done, a third warning happens. The system takes control of the car, including decelerating, and this situation can only be solved by looking at the road. After a driver looks back on the road, the assistance system enters a non-hands-free-driving state. This takeover procedure can also happen at any time if an invalid situation for the system occurs and the eyes are not focused on the road. If conflicts are resolved before a third warning, the car continues driving hands-free without any warning.

This textual description is translated into a Petri net to compare this model and the discovered process models. To convert this model into a Petri net, we assigned each state a place and a marking, as shown in [12]. The resulting Petri net is depicted in Fig. 4. We use τ-transitions since the reasons for starting the takeover can not properly be defined using a single transition and, therefore, can lead to an unreadable model. Moreover, after consulting domain experts, we introduced multiple transitions to reveal the input that drivers perform to turn the hands-free driving mode off. By executing "State 1", "State 2", or "State 3", hands-free driving is disabled, leading to place *p5*. The first conflict state is entered with "Warning 1" (*p2*). When the conflict is resolved, the system is again in the hands-free driving state (*p1*). When the conflict is not resolved, and after a certain period of time, the conflict escalates by executing "Warning 2", leading to place *p3*. Again, the conflict

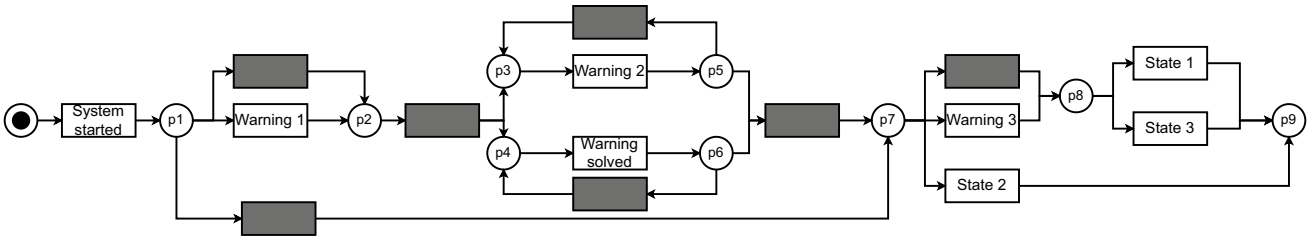


Fig. 6 Petri net discovered by applying the Inductive Miner infrequent [28] with a threshold of 0.2

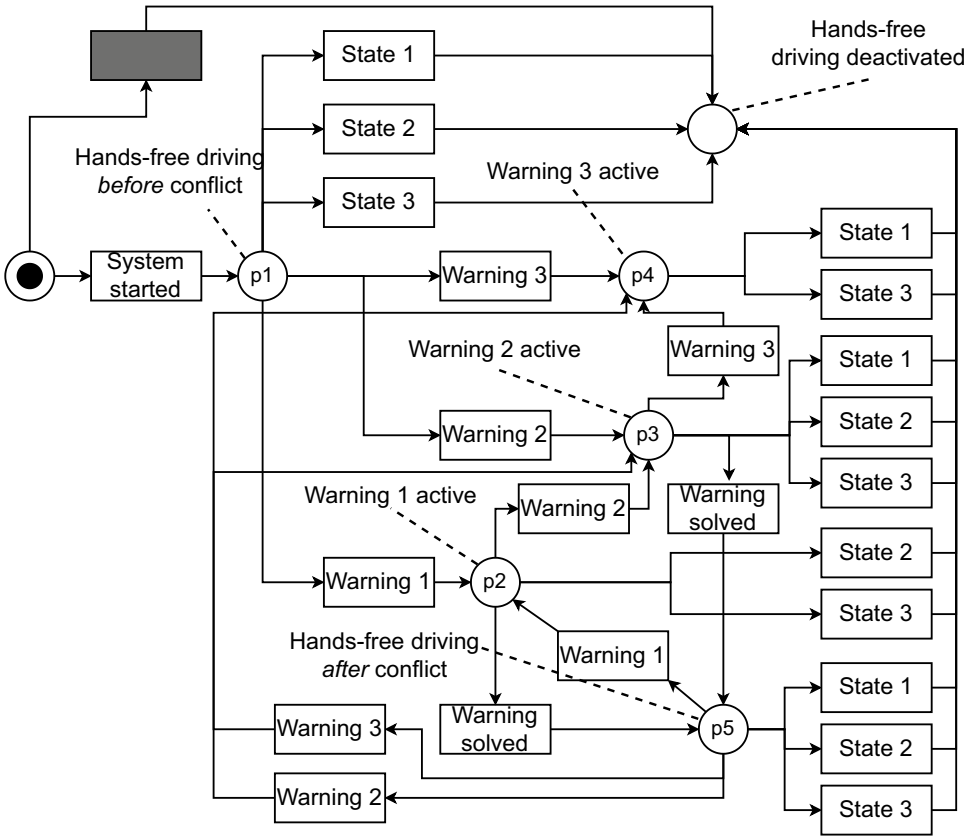


Fig. 7 Petri net discovered by applying directly follows mining [30]

can be resolved; however, when the conflict is not resolved, "Warning 3" is executed, leading to place *p4*. When "Warning 3" is executed, only "State 3" can be executed next, leading to the turn-off of the system and place *p5*. "Warning 3" can also be executed in other system states.

Discovering De-facto Models

In contrast to constructing a Petri net from an existing description, we apply process-discovery techniques to the event log to discover de-facto models. To discover process models, we apply several process discovery algorithms to the transformed event log, all implemented in ProM [16]. In the following, we provide an overview of the approaches and showcase a selection of Petri nets.

We employed variations of the α algorithm, including the classic α algorithm [17], the α^{+} algorithm [18], the α^{++} algorithm [19], and the $\alpha^{\#}$ algorithm [20]. However, process models discovered by these techniques have issues, as later shown in the evaluation of models. Most of them are not a workflow net (WF-net) or unsound, i.e., they are not of use.

Also, we use region theory [21, 22] by applying the region miner [23–27]. In this process, we first mine a transition system by using the event name as a backward key

Table 4 Conformance checking results for the de-jure model and models discovered by applying various algorithms and filters

Algorithm	Filter parameter	WF-net	Sound	Fitness	Precision	Generalization	Simplicity	F-1 score
α	–	No	–	–	–	–	–	–
α^{+}	–	Yes	No	–	–	–	–	–
α^{++}	–	Yes	No	–	–	–	–	–
$\alpha^{\#}$	–	No	–	0.72	0.57	0.97	1.00	0.64
Region miner	–	Yes	Yes	1.00	0.94	0.85	0.50	0.97
Inductive miner infrequent	0.0	Yes	Yes	1.00	0.68	0.97	0.66	0.81
Inductive miner infrequent	0.1	Yes	Yes	0.99	0.82	0.97	0.64	0.90
Inductive miner infrequent	0.2	Yes	Yes	0.99	0.82	0.97	0.64	0.90
Inductive miner infrequent	0.3	Yes	Yes	0.89	0.88	0.97	0.70	0.88
Inductive miner infrequent	0.4	Yes	Yes	0.89	0.88	0.97	0.70	0.88
Inductive miner infrequent	0.5	Yes	Yes	0.89	0.88	0.97	0.70	0.88
Inductive miner infrequent	0.6	Yes	Yes	0.89	0.88	0.97	0.70	0.88
Inductive miner infrequent	0.7	Yes	Yes	0.89	0.88	0.97	0.70	0.88
Inductive miner infrequent	0.8	Yes	Yes	0.89	0.88	0.97	0.70	0.88
Inductive miner infrequent	0.9	Yes	Yes	0.68	0.79	0.97	0.73	0.73
Inductive miner infrequent	1.0	Yes	Yes	0.51	1.00	0.97	0.82	0.68
Directly follows miner	0.0 (paths)	Yes	Yes	–	–	–	–	–
Directly follows miner	0.1 (paths)	Yes	Yes	0.47	1.00	0.66	1.00	0.64
Directly follows miner	0.2 (paths)	Yes	Yes	0.47	1.00	0.66	1.00	0.64
Directly follows miner	0.3 (paths)	Yes	Yes	0.47	1.00	0.66	1.00	0.64
Directly follows miner	0.4 (paths)	Yes	Yes	0.47	1.00	0.66	1.00	0.64
Directly follows miner	0.5 (paths)	Yes	Yes	0.47	1.00	0.66	1.00	0.64
Directly follows miner	0.6 (paths)	Yes	Yes	0.66	1.00	0.74	0.78	0.80
Directly follows miner	0.7 (paths)	Yes	Yes	0.66	1.00	0.74	0.78	0.80
Directly follows miner	0.8 (paths)	Yes	Yes	0.91	1.00	0.86	0.68	0.95
Directly follows miner	0.9 (paths)	Yes	Yes	0.93	1.00	0.87	0.64	0.96
Directly follows miner	1.0 (paths)	Yes	Yes	1.00	0.94	0.83	0.47	0.97
De-Jure model	–	Yes	Yes	0.91	0.99	0.88	0.62	0.95

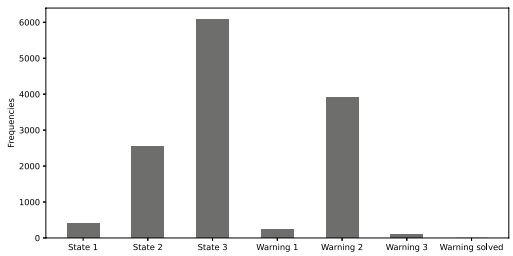
Results are rounded to the second decimal

and a set abstraction of size one. Moreover, self-loops were removed, and label-splitting was performed. By using the mentioned methods, a Petri net is created. The resulting Petri net is displayed in Fig. 5.

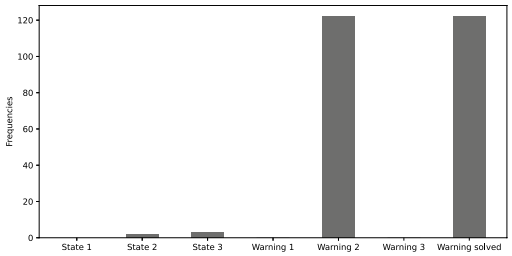
Each place of the Petri net corresponds to the system's states. The hands-free driving state is in place *p1*. All states and warnings can happen there. However, the execution of "Warning 2" should not be possible according to the definition. Place *p2* is the state in which "Warning 1" is active. The warning can be resolved, which leads to *p1*. Also, the system can be turned off by utilizing "State 2" and "State 3", which was not part of the description but is possible according to domain experts. Additionally, the conflict can be escalated via "Warning 2" to *p3*. In *p3*, the system can be turned off, the warning can be solved (returning to *p1*), or the conflict can escalate by executing "Warning 3" leading to *p4*. In *p4*, the system's next action is to turn itself off. However, in contrast to the description, two options for the turn-off are possible in the given data.

We also applied the Inductive Miner infrequent (IMf) [28], which is based on the Inductive Miner [29], the state-of-the-art process-discovery algorithm. The resulting Petri net when applying the IMf with a threshold of 0.2 is displayed in Fig. 6.

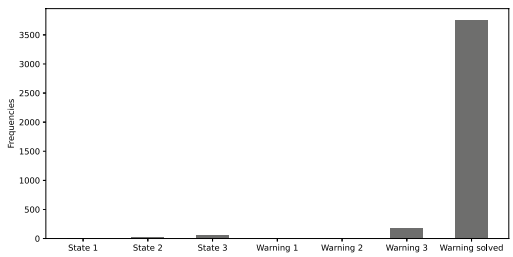
This Petri net shows the escalation of warnings and all states that lead to the turn-off of the system. In contrast to previous models, this model does not have multiple transitions with the same activity. However, this model has some issues. First, "Warning solved" can be infinitely often executed in a row, which does not make much sense given the domain knowledge and the knowledge about the data. Second, "Warning solved" can be executed if no warning happens. Third, in contrast to the de-facto model (see Fig. 4) and the model discovered by region-based mining (see Fig. 5), the markings are less interpretative. For instance, if *p2* is marked, one must check the data to observe whether "Warning 1" has been executed. These issues are based on how the algorithm works.



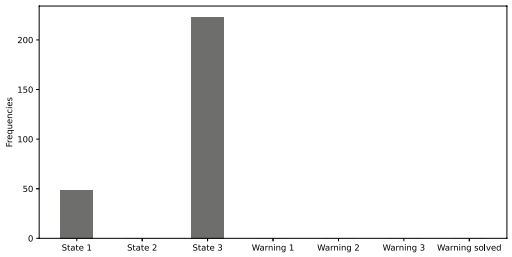
(a) Frequency distribution for being in hands-free driving with no warning ($p1$).



(b) Frequency distribution for having "Warning 1" ($p2$).



(c) Frequency distribution for having "Warning 2" ($p3$).



(d) Frequency distribution for having "Warning 3" ($p4$).

Fig. 8 Distributions of the next executed activity for a selection of places shown in the Petri net portrayed in Fig. 5

We also applied a directly follows-based approach using [30]. We set the activity parameter to 1.0 and changed the path parameter. The resulting Petri net by setting the path parameter to 1.0 is depicted in Fig. 7.

The result is similar to the process model discovered by the region miner (see Fig. 5). Each marking corresponds to a system's state. The major difference is that the model discovered by the directly follows miner distinguishes between the initial hands-free driving ($p1$) and the hands-free driving after conflict resolution ($p5$). However, both places have the same labeled outgoing transitions. Thus, the differentiation leads to a more complex model, while the insights concerning the control flow are the same. Another difference is the option for an empty trace, which is unrelated to showing how the systems operate.

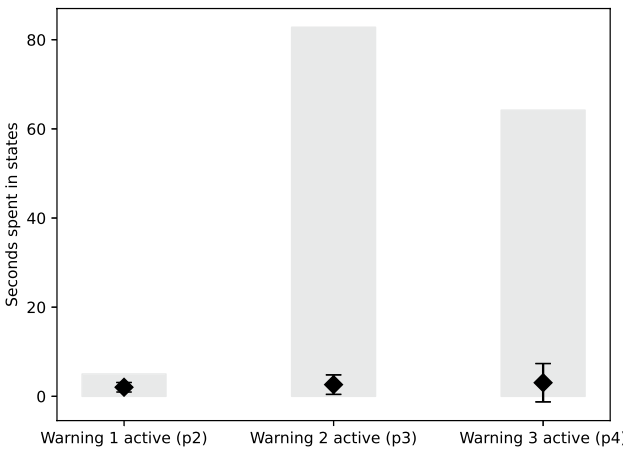
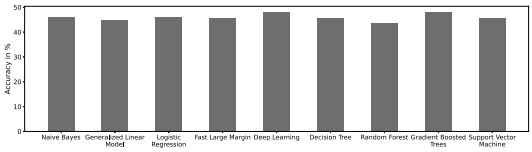


Fig. 9 Time spent is selected states of the Petri net depicted in Fig. 5. The bars show the minimum and maximum value, the diamonds the corresponding mean, and the whiskers represent the standard deviation

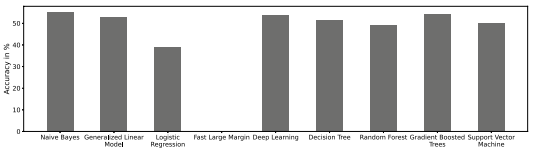
Evaluation of the Different Models

After creating a de-jure model by utilizing domain knowledge and discovering various process models, we evaluate them to find the most representative process model. In particular, we measure fitness by utilizing alignments with a standard cost function [31], computed precision [32] and generalization [33] scores, as well as the simplicity of the model [34]. In addition, we computed F1 scores using the obtained fitness and precision scores. Besides, we used WOFLAN [35] to check soundness and whether the Petri net is a WF-net. The results are depicted in Table 4.

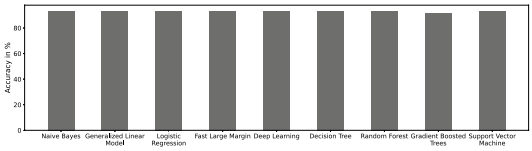
As denoted in Table 4, the de-jure model explains the underlying data well since the F1 score is 0.95. This shows that the system performs most of the time as designed by the experts. Comparable models are found by using the region-based approach (F1 score of 0.97) and the directly follows miner (F1 score of 0.97). The results of the different



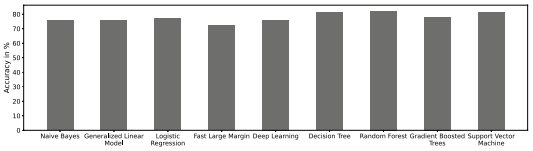
(a) Accuracy of predicting the next action being in hands-free driving (*p1*).



(b) Accuracy of predicting the next action for having "Warning 1" (*p2*).



(c) Accuracy of predicting the next action for having "Warning 2" (*p3*).



(d) Accuracy of predicting the next action for having "Warning 3" (*p4*).

Fig. 10 Accuracy measurements for predicting the next activity in selected places of the Petri net portrayed earlier in Fig. 5

α miners are not of use due to either missing soundness or not being a WF-net. Applying the Inductive Miner leads, as shown, to unsatisfying models that lack interpretability. The region-based approach produces a slightly better F1 score than the de-jure model by having a perfect fitness score. In addition, this model has better scores concerning generalization and simplicity than the model obtained by applying the directly follows miner. Due to the better scores and by considering domain experts' opinions, we chose the region-based model as our representative process model.

Conformance Analysis

As described before, the model discovered by the region miner has perfect fitness. When aligning the model (see Fig. 5) and the event log, we get a comprehensive overview of the frequency of the execution of the transitions. Consequently, we are also getting insights into users' behavior. An overview is depicted in Fig. 8.

As shown in Fig. 8, the choice of the next activity varies, sometimes based on the system itself. For instance, for *p1* and *p4*, "Warning solved" cannot be executed.

For *p1* (Fig. 8a), "State 3" is the most executed activity ($\approx$ 46% of the time). The combination of the frequencies for "State 1" and "State 2" happens less than for "State 3". This can be related to the user behavior, perhaps the preferable turn-off option. The second most activity is "Warning 2". In contrast to "Warning 2", "Warning 1" happens rarely. This is a contradiction to the requirements. Given these requirements, there is an escalation order. However, as we denote, the second-level warning usually happens. Possible causes are a data logging issue. Nevertheless, more investigations are needed.

For *p2* (Fig. 8b), "Warning 2" and "Warning solved" happen the most and equally often (each $\approx$ 49% of the time). This means that roughly half of the time, a driver manages to resolve the conflict, and half of the time, the conflict escalates to the next level. However, sometimes drivers turn the feature off ("State 2" and "State 3"). "State 1" and "Warning 3" never happen.

For *p3* (Fig. 8c), the warning is often resolved ($\approx$ 93% of the time). Sometimes, the conflict escalated to the third level. Besides, the feature is occasionally turned off by drivers.

For *p4* (Fig. 8d), there is only the choice between "State 1" and "State 3". As shown, the latter happens four times more often than the former ($\approx$ 82% of the time).

In general, this analysis shows that "State 3" is a highly frequent activity in all places in the model. Moreover, we observed that there are potential logging issues concerning the escalation of warnings, which need to be investigated by domain experts.

Performance Analysis

We conduct a performance analysis based on the region-based process model (depicted in Fig. 5). The performance analysis reveals how much time is spent in places of a Petri net and how much time passed before firing a transition. The time spent in the warning places, i.e., *p2*, *p3*, and *p4*, are highly interesting to further check the system requirements. We use the package provided in ProM [16] for this analysis. The results are displayed in Fig. 9.

As shown in Fig. 9, the maximum times vary, and, consequently, the overall distribution. While not much time is spent in *p2*, over a minute is potentially spent in *p3* and *p4*.

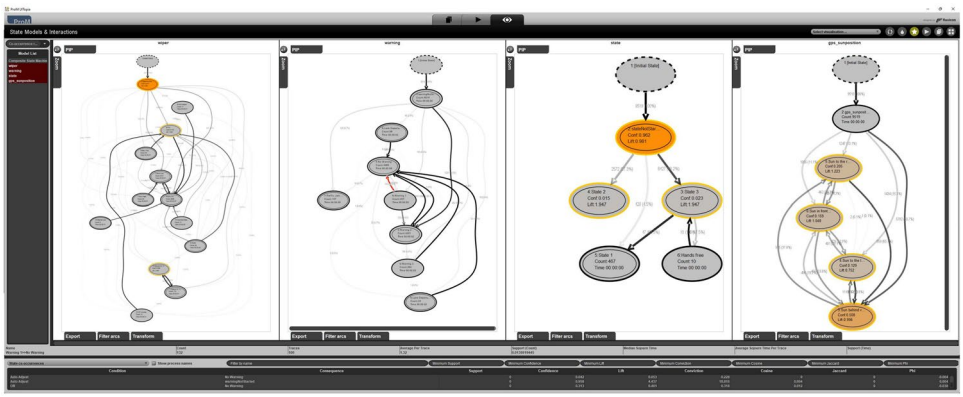


Fig. 11 Interface of the CSMM [39]. By clicking on the arcs between states, we can observe co-occurring states, indicated by their color

However, the average values are close to each other, and the deviations are minor. The time for "Warning 1" being active is dense. For "Warning 2" it is a bit less dense; however, there are potential outliers as shown by the standard deviation. "Warning 3" is the less dense distribution, but still, it is not as bad as the outliers may indicate. Possible reasons for that are logging issues, which can occur. Besides the outliers, the systems seem to operate as intended.

Decision Mining

After measuring the time spent in selected system states, we investigate reasons for certain paths. The technique which enables us to do so is called decision mining. First, we identify decision points in the process, i.e., places with more than one outgoing arc in our picked process model (see Fig. 5). In our case, these are *p1*, *p2*, *p3*, and *p4*. By aligning the event log with the data, we use the different attributes of an event to predict the next activity. More information can be found in [36]. In this process, we create a supervised learning problem. For each decision point, we extract a dataset by using PM4Py [37]. Each dataset is used as input for RapidMiner.[1] In RapidMiner, we use the auto-model functionality to predict the next activity for each decision point. Using auto-modeling in RapidMiner, different techniques with various parameter settings are applied to the data. The results are showcased in Fig. 10.

The result concerning *p1* is displayed in Fig. 10a. The accuracy values are similar, while the highest result is achieved by using deep learning (48%). The accuracy value is greater than the probability of always picking the most popular value (≈ 46%). However, the accuracy is not satisfying. As a result, the next action when being in place *p1* can not be well predicted with our data.

The result concerning *p2* is presented in Fig. 10b. The accuracy values are often similar. The accuracy value for fast large margin could not be computed and returned an error, and logistic regression performs worse than others. The probability for the most prominent next activity is ≈ 49%, while the greatest accuracy score is 55.1%, achieved by Naive Bayes. Therefore, the accuracy is better than the majority vote, but it suffers simultaneously for being too low.

The result concerning *p3* is shown in Fig. 10c. As we can denote, the accuracy values are always above 90%. When comparing this with the probability of the most frequent activity, which is ≈ 93%, we conclude that our classifiers perform not better than the majority vote.

The result concerning *p4* is depicted in Fig. 10d. There are various accuracy values, ranging from ≈ 72% to ≈ 82%. As described earlier, the probability for the most common activity is ≈ 82%. Consequently, the next activity cannot be well predicted.

In general, we observe that our classifiers often do not outperform majority votes. A potential reason is that our

Table 5 Overview of selected results from applying the CSSM [39]

From	To	Affected feature and state	Confidence	Lift
Initial warning	Warning 1	Sun: Right	0.19	1.24
Initial warning	Warning 3	Sun: Right	0.20	1.30
Warning 1	Warning 2	Sun: Left	0.22	1.26
Warning 2	Warning 3	Sun: Front	0.15	1.30
No warning	Warning 1	Sun: Front	0.20	1.84
No warning	Warning 1	Sun: Left	0.20	1.50
No warning	Warning 3	Sun: Behind	0.80	1.37
No warning	Warning 3	Wiper: Auto off	0.10	2.30
Initial state	State 1	Sun: Left	0.21	1.38
Initial state	State 1	Wiper: Auto off	0.28	8.10

Values are rounded to the second decimal

[1] https://rapidminer.com/.

data are missing information, for example, a vehicle's speed. Also, we do not know the state of the environment, which may have an impact. Potential environmental effects include traffic jams, temperature, road conditions, software versions, etc. As a result, we have limited knowledge about the situation when a decision occurs and, therefore, cannot predict well. Even though we cannot predict the next action well, the result shows that the feature we considered does not cause the next action, indicating that there are no major design issues.

Multi-perspective Analysis

In this section, we provide a multi-perspective analysis of the process. First, we present the motivation and the chosen approach. Second, we describe the data transformation to utilize this tool. Third, we provide the results from our analysis.

So far, we have concentrated on the control-flow perspective of the different states and warnings of the system. However, a vehicle consists of many subsystems which interact with each other. Also, the environment influences the system. These dynamics are captured in the transmitted vehicle data. In the following, we take a closer look at these dynamics. The domain experts are interested in the relationship between the sun's relative position, the system's states and warnings, and the wiper state. For our analysis, we use the Composite State Machine Miner (CSMM) [38, 39]. The different values of a feature are interpreted as its states. The CSMM enables us to observe the dynamics between the states of a feature individually. Additionally, the tools allow us to check which states across the features occurred together by returning a lift and confidence value. By utilizing this technique, we are interested in observing links across the features and indicators for certain behaviors. For example, whether a sun's state often co-occurs with a vehicle's warning state which may indicate potential issues.

For this task, we preprocessed the data as we did earlier. However, to use the tool, we must transform our data differently than before. Besides, a drawback of the CSMM is that it assumes that initial states are the same. Thereby, it only tracks changes. However, our vehicle data does not always start in the same state. For example, the relative position of the car to the sun can be different for each run. Since, according to the experts, the sun's position may influence the system, we create an initial state for the sun, which then directly changes to the real initial state. This enables us to analyze the sun's influence compactly and comprehensively. A screenshot of the tool that loaded our data is provided in Fig. 11.

For our results, we focus on relationships that have a minimum confidence value of 0.1 and a minimum lift value of 1.2. We focus on the switch between states, i.e., the arcs connecting them. Our results are shown in Table 5.

As shown in Table 5, most confidence values are between 0.1 and 0.3. Such findings indicate that the corresponding findings are questionable. Also, the findings reveal that the assumed influence does not exist. The most striking finding deals with "No Warning" and "Warning 3". With a confidence value of 0.8 and a lift value of 1.37, this finding shows a positive association. However, considering that a lift value nearly to one indicates independence, the impact of our finding is limited.

In summary, we applied this analysis to capture the dynamics within the systems and to observe co-occurring behavior. By doing so, we checked whether the assumption that the environment influences the system holds. Our analysis shows that there are some influences. However, the corresponding confidence and lift values are often low. This leads to the conclusion that the assumed influences do not exist, so further investigations are needed.

Conclusion

In this section, we summarize our work and provide pointers for future work.

In this work, we presented in detail how to enrich and transform continuous data recorded by cyber-physical systems into an event log. We analyzed the event log, and by conducting a small variant analysis, we found out that the most frequent five variants are responsible for nearly 90% of the behavior. By looking at the cases' throughput time, we found out that most engagements with the features are less than four minutes long. Furthermore, we applied process-discovery and conformance-checking techniques to the transformed event log. Also, we created a de-jure model based on the feature's documentation. By utilizing the conformance scores and domain experts' opinions, we picked a well-representative model. This model revealed differences between the documentation of the system and the actual behavior. According to experts, the behavior was as designed in most aspects, but some behaviors provide points for further investigation, for instance, the warning escalation. At the same time, critical aspects of the feature work as intended. Additionally, we applied process-enhancement techniques by utilizing our picked model. We obtained the behavior that drivers tend to have when they are in different states of the system. We also checked the amount of time spent in a warning state. However, the results have to be treated with caution due to logging issues. By using decision mining, we aimed to discover the reasons behind the actions taken in the system. However, most of the time, the accuracy of our classifiers was roughly the same as when using majority voting. We also aimed to capture a multi-perspective

view of the system to discover which states often occur. For this multi-perspective view, we not only focused on states and warnings, but we also took the wiper state and the relative position between the sun and a vehicle into account. Most of our reported results lacked a high confidence value, limiting the reasoning of our findings. At the same time, we showed with the analysis that there are no flaws in the system based on the available data and the considered features.

There are points for future work. As pointed out, the accuracy values of the classifiers are unsatisfying. Therefore, one cannot easily interpret the reasons for decisions. However, it is possible that other attributes can help us in understanding the reasons. For example, a vehicle's speed or the information of whether a curve is driven or whether a vehicle is in a traffic jam helps to capture the situation of a vehicle. Also, inter-case attributes can be interesting, for example, the number of engagements during a journey until that run. Besides the need for more attributes, there is a need for better techniques concerning a multi-perspective view. As pointed out, the technique suffers from only capturing state changes and assigning different states the same initial state. As a result, complex systems with changing conditions, both initial and during the execution, like a vehicle, suffer from this approach since data have to be artificially adjusted. For a data-driven analysis, we need to invent techniques that capture that.

Acknowledgements This research is supported by the Ford-RWTH Aachen University Alliance program. We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research. Grant number 1191945.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability The data is not publicly available.

Declarations

Conflict of interest Harry H. Beyel works on a project financed by Ford. Omar Makke and Oleg Gusikhin are Ford employees. Wil M. P. van der Aalst works part-time for Celonis.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Lee EA. Cyber physical systems: design challenges. In: IEEE (ISORC). 2008. p. 363–9. https://doi.org/10.1109/ISORC.2008.25.
2. Gubbi J, Buyya R, Marusic S, Palaniswami M. Internet of things (iot): a vision, architectural elements, and future directions. Future Gener Comput Syst. 2013;29(7):1645–60. https://doi.org/10.1016/j.future.2013.01.010.
3. van der Aalst WMP. Process mining–data science in action. 2nd ed. 2016. https://doi.org/10.1007/978-3-662-49851-4.
4. Janiesch C, Koschmider A, Mecella M, Weber B, Burattin A, Di Ciccio C, Fortino G, Gal A, Kannengiesser U, Leotta F, Mannhardt F, Marrella A, Mendling J, Oberweis A, Reichert M, Rinderle-Ma S, Serral E, Song W, Su J, Torres V, Weidlich M, Weske M, Zhang L. The internet of things meets business process management: a manifesto. IEEE Syst Man Cybern Mag. 2020;6(4):34–44. https://doi.org/10.1109/MSMC.2020.3003135.
5. Keates O. Integrating iot with BPM to provide value to cattle farmers in Australia. In: Business process management workshops. 2019. p. 119–29. https://doi.org/10.1007/978-3-030-37453-2_11.
6. van Eck ML, Sidorova N, van der Aalst WMP. Enabling process mining on sensor data from smart products. In: Tenth IEEE international conference on Research Challenges in Information Science, RCIS 2016, Grenoble, France, June 1–3, 2016. p. 1–12. https://doi.org/10.1109/RCIS.2016.7549355.
7. Koschmider A, Janssen D, Mannhardt F. Framework for process discovery from sensor data. In: EMISA. 2020. p. 32–8.
8. Astromskis S, Janes A, Mairegger M. A process mining approach to measure how users interact with software: an industrial case study. In: ICSSP. 2015. p. 137–41. https://doi.org/10.1145/2785592.2785612.
9. Makke O, Gusikhin O. Robust iot based parking information system. In: Smart cities, green technologies, and intelligent transport systems. 2021. p. 204–27. https://doi.org/10.1007/978-3-030-89170-1_11.
10. Beyel HH, Makke O, Yuan F, Gusikhin O, van der Aalst WMP. Analyzing cyber-physical systems in cars: a case study. In: Proceedings of the 12th International Conference on Data Science, Technology and Applications, DATA 2023, Rome, Italy, July 11–13, 2023. p. 195–204. https://doi.org/10.5220/0012136000003541.
11. Beyel HH, Makke O, van der Aalst Gusikhin O, WMP. Analyzing behavior un cyber-physical systems in connected vehicles: a case study. Business Process Management Workshops—BPM 2023 International Workshops. Utrecht, The Netherlands:; September 11–15, 2023. p. 92–104. https://doi.org/10.1007/978-3-031-50974-2_8.
12. Reisig W. Petri nets: an introduction. EATCS monographs on theoretical computer science, vol. 4. 1985. https://doi.org/10.1007/978-3-642-69968-9.
13. Carmona J, van Dongen BF, Solti A, Weidlich M. Conformance checking—relating processes and models. 2018. https://doi.org/10.1007/978-3-319-99414-7.
14. Reda I, Andreas A. Solar position algorithm for solar radiation applications. Sol Energy. 2004;76(5):577–89. https://doi.org/10.1016/j.solener.2003.12.003.
15. Holmgren WF, Hansen CW, Mikofski MA. pvlib python: a python package for modeling solar energy systems. J Open Source Softw. 2018;3(29):884. https://doi.org/10.5281/zenodo.8368494.
16. van Dongen BF, Medeiros AKA, Verbeek HMW, Weijters AJMM, van der Aalst WMP. The prom framework: a new era in process

mining tool support. In: Applications and theory of Petri nets. 2005. p. 444–54. https://doi.org/10.1007/11494744_25.
17. van der Aalst WMP, Weijters T, Maruster L. Workflow mining: discovering process models from event logs. IEEE Trans Knowl Data Eng. 2004;16(9):1128–42. https://doi.org/10.1109/TKDE.2004.47.
18. Medeiros AKA, van Dongen BF, van der Aalst WMP, Weijters AJMM. Process mining for ubiquitous mobile systems: an overview and a concrete algorithm. In: Baresi L, Dustdar S, Gall HC, Matera M, editors. Ubiquitous mobile information and collaboration systems, second CAiSE workshop, UMICS 2004, Riga, Latvia, June 7–8, 2004, Revised Selected Papers. Lecture Notes in Computer Science, vol. 3272. 2004. p. 151–65. . https://doi.org/10.1007/978-3-540-30188-2_12.
19. Wen L, van der Aalst WMP, Wang J, Sun J. Mining process models with non-free-choice constructs. Data Min Knowl Discov. 2007;15(2):145–80. https://doi.org/10.1007/S10618-007-0065-Y.
20. Wen L, Wang J, van der Aalst WMP, Huang B, Sun J. Mining process models with prime invisible tasks. Data Knowl Eng. 2010;69(10):999–1021. https://doi.org/10.1016/J.DATAK.2010.06.001.
21. Ehrenfeucht A, Rozenberg G. Partial (set) 2-structures. Part I: basic notions and the representation problem. Acta Inform. 1990;27(4):315–42. https://doi.org/10.1007/BF00264611.
22. Ehrenfeucht A, Rozenberg G. Partial (set) 2-structures. Part II: state spaces of concurrent systems. Acta Inform. 1990;27(4):343–68. https://doi.org/10.1007/BF00264612.
23. van der Aalst WMP, Rubin VA, Verbeek HMW, van Dongen BF, Kindler E, Günther CW. Process mining: a two-step approach to balance between underfitting and overfitting. Softw Syst Model. 2010;9(1):87–111. https://doi.org/10.1007/s10270-008-0106-z.
24. Cortadella J, Kishinevsky M, Lavagno L, Yakovlev A. Deriving petri nets for finite transition systems. IEEE Trans Comput. 1998;47(8):859–82. https://doi.org/10.1109/12.707587.
25. Solé M, Carmona J. Light region-based techniques for process discovery. Fundam Inform. 2011;113(3–4):343–76. https://doi.org/10.3233/FI-2011-612.
26. Solé M, Carmona J. Incremental process discovery. Trans Petri Nets Other Model Concurr. 2012;5:221–42. https://doi.org/10.1007/978-3-642-29072-5_10.
27. Solé M, Carmona J. Region-based foldings in process discovery. IEEE Trans Knowl Data Eng. 2013;25(1):192–205. https://doi.org/10.1109/TKDE.2011.192.
28. Leemans SJJ, Fahland D, van der Aalst WMP. Discovering block-structured process models from event logs containing infrequent behaviour. In: Lohmann N, Song M, Wohed P, editors. Business Process Management Workshop—BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers. Lecture Notes in Business Information Processing, vol. 171. 2013. p. 66–78. https://doi.org/10.1007/978-3-319-06257-0_6.
29. Leemans SJJ, Fahland D, van der Aalst WMP. Discovering block-structured process models from event logs—a constructive approach. In: Colom JM, Desel J, editors. Application and theory of Petri nets and concurrency—34th International Conference, PETRI NETS 2013, Milan, Italy, June 24–28, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7927. 2013. p. 311–29. https://doi.org/10.1007/978-3-642-38697-8_17.
30. Leemans SJJ, Poppe E, Wynn MT. Directly follows-based process mining: exploration & a case study. In: International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24–26, 2019, p. 25–32. https://doi.org/10.1109/ICPM.2019.00015.
31. Adriansyah A, Sidorova N, van Dongen BF. Cost-based fitness in conformance checking. In: Caillaud B, Carmona J, Hiraishi K, editors. 11th international conference on Application of Concurrency to System Design, ACSD 2011, Newcastle Upon Tyne, UK, 20–24 June, 2011. p. 57–66. https://doi.org/10.1109/ACSD.2011.19.
32. Adriansyah A, Munoz-Gama J, Carmona J, van Dongen BF, van der Aalst WMP. Measuring precision of modeled behavior. Inf Syst E Bus Manag. 2015;13(1):37–67. https://doi.org/10.1007/s10257-014-0234-7.
33. Buijs JCAM, van Dongen BF, van der Aalst WMP. Quality dimensions in process discovery: the importance of fitness, precision, generalization and simplicity. Int J Coop Inf Syst. 2014. https://doi.org/10.1142/S0218843014400012.
34. Weerdt JD, Backer MD, Vanthienen J, Baesens B. A critical evaluation study of model-log metrics in process discovery. In: Muehlen M, Su J, editors. Business Process Management Workshops—BPM 2010 International Workshops and Education Track, Hoboken, NJ, USA, September 13–15, 2010, Revised Selected Papers. Lecture Notes in Business Information Processing, vol. 66. 2010. p. 158–69. https://doi.org/10.1007/978-3-642-20511-8_14.
35. Verbeek HMW, Basten T, van der Aalst WMP. Diagnosing workflow processes using woflan. Comput J. 2001;44(4):246–79. https://doi.org/10.1093/comjnl/44.4.246.
36. Leoni M, van der Aalst WMP, Dees M. A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Inf Syst. 2016;56:235–57. https://doi.org/10.1016/j.is.2015.07.003.
37. Berti A, van Zelst S, Schuster D. Pm4py: a process mining library for python. Softw Impacts. 2023;17:100556. https://doi.org/10.1016/j.simpa.2023.100556.
38. van Eck ML, Sidorova N, van der Aalst WMP. Discovering and exploring state-based models for multi-perspective processes. In: Rosa ML, Loos P, Pastor O, editors. Business Process Management—14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18–22, 2016. Proceedings. Lecture Notes in Computer Science, vol. 9850. 2016. p. 142–57. https://doi.org/10.1007/978-3-319-45348-4_9.
39. van Eck ML, Sidorova N, van der Aalst WMP. Composite state machine miner: discovering and exploring multi-perspective processes. In: Azevedo L, Cabanillas C, editors. Proceedings of the BPM Demo Track 2016 co-located with the 14th International Conference on Business Process Management (BPM 2016), Rio de Janeiro, Brazil, September 21, 2016. CEUR Workshop Proceedings, vol. 1789. 2016. p. 73–7. https://ceur-ws.org/Vol-1789/bpm-demo-2016-paper14.pdf.