



Improving conformance checking in process modelling: a multiperspective algorithm

Rui Calheno¹ · Paulo Carvalho¹ · Solange Rito Lima¹ · Pedro Rangel Henriques¹ · Mateo Ramos Merino²

Accepted: 15 April 2023 / Published online: 13 May 2023
© The Author(s) 2023

Abstract

Business process model and notation (BPMN) is a popular notation used for process modelling mainly due to its high expressiveness. However, BPMN has shortcomings when dealing with specific domains (namely Hazard Analysis and Critical Control Points systems), struggling to model activity duration, quality control points, activity effects and monitoring nature. To tackle these limitations, the business process model and notation extended expressiveness (BPMN-E²) was proposed. In this paper, a multiperspective conformance checking algorithm is developed focusing on detecting non-conformity between an event log and a process model, regarding the information provided by the new elements within BPMN-E². The proposed algorithm follows a two-step approach that starts by converting the model into a *directly follows model* (annotated with conformance rules), which is then used in a second phase to perform conformance checking effectively. This modular approach allows to apply the proposed algorithm to other process model notations than BPMN-E². An event log clustering technique was also developed to downsize large-event logs without compromising data relevance. In this way, both the multiperspective algorithm and the log-downsize clustering technique here proposed are a key contribution to improve conformance checking in process modelling, as evinced in the proof-of-concept provided.

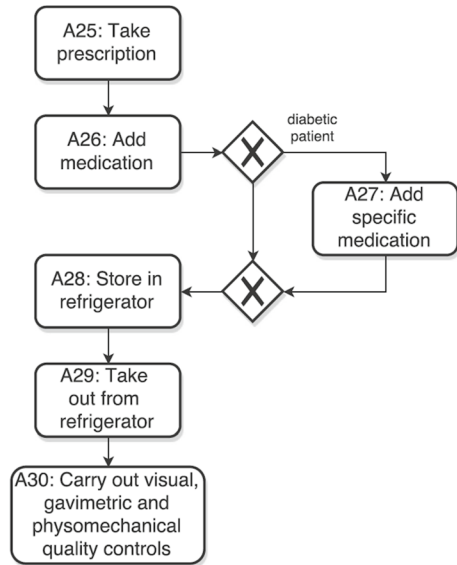
Keywords BPMN · BPMN-E² · Conformance checking · Process modelling

1 Introduction

The widespread use of Information and Communications Technologies in people and organisations activities has led to a massive increase in the amount of data generated by end systems. These data play a key role and added value within the new business paradigm that organisations face. Currently, organisations aim at extracting information and value from data stored in their information systems to better

Extended author information available on the last page of the article

Fig. 1 BPMN documentation excerpt of PN mixtures elaboration process [7]



improve their business and gain a competitive advantage over other organisations [1–3].

Process mining is an approach that combines data mining and process modelling to properly analyse event data. The aim is to discover, monitor and improve real processes through the extraction of meaningful insights and knowledge from event logs [4, 5]. For this, process mining relies on techniques that can be grouped in distinct areas considering its intended purpose [6]: (a) *Discovery*, given an event log, produces a process model; (b) *Conformance Checking*, taking an event log and a process model, produces a report comparing these components and identifying possible non-conformity between them; and (c) *Enhancement*, given both an event log and a process model, improves the latter with new information recorded in the former.

Attending to the extensive use of business process model and notation (BPMN) in a vast scope of business applications, business process model and notation extended Expressiveness (BPMN-E²) has been developed as an extension of its functionality. BPMN-E² gives to the process modelling designer the possibility to describe in a more detailed way the workflow behaviour, the activities being performed and the context of one particular process [7]. The need for an enriched context-aware approach is even more evident when dealing with process control environments, such as *Hazard Analysis and Critical Control Points* (HACCP), where it is vital to ensure the safety and security of products manufactured in a given industry (e.g. food, pharmaceutical or cosmetics) [7]. In these scenarios, the business processes are explained in detail using both a BPMN diagram and a natural language description that explains the activity behaviour, quality controls or how to proceed in case of hazards. An example of such a process description can be found in Figs. 1 and 2.

Jointly, a BPMN model and a natural language description provide a rich and complete representation of the business process; however, only the model can be

Fig. 2 Natural language documentation excerpt of PN mixtures elaboration process [7]

In this phase, it is necessary to read the prescription and the note of all relevant aspects (diabetic, [...]). Then, the pharmaceutical operator must add now the prescribed medicaments for this type of parenteral nutrition. First, add 100 mg of ranitidine, [...] and remove the mixture during 30 seconds. Second, if the patient is diabetic, add the units of U-100 insulin pointed by the prescription. Next, store the mixture in the refrigerator. One hour after, take out the parenteral nutrition from the refrigerator and measure its temperature. Some quality controls must be carried in order to validate the correct elaboration of the mixture. First, you have to perform a visual control by checking if the closure of the PN bag is ok. Then, check the temperature measured in A28, it must be between 2 and 8C. Finally, check the physicochemical characteristics by measure the pH. It must be between 5.4 and 6.5

used during process mining tasks, which results in a loss of valuable information [7]. Furthermore, human auditors should study in detail both documents, namely the BPMN and the one expressed in natural language, which could over complicate auditing tasks. With this in mind, the authors of the BPMN-E² notation aimed to extend the BPMN element set with new stereotypes that represent the most valuable information, formerly only present in natural language documents.

In this paper, we propose the development of a new conformance checking algorithm, that takes into account the extended expressiveness that the BPMN-E² notation offers. This data-aware multiperspective algorithm allows for a richer process analysis, providing insights into *activity duration*, *activity effects*, and *decision points* related non-conformities. Furthermore, it takes advantage of BPMN-E² distinction of monitored and non-monitored activities to provide reliable conformance results by appropriate filtering. This is expected to reduce possible false-negative conformance errors from being detected when working with partially monitored environments. To prove the usefulness of the proposed algorithm and support the BPMN-E² notation, a new python library is built on top of an existing process mining library (PM4Py), and used in distinct test scenarios. Note that, despite being developed with BPMN-E² and HACCP systems in mind, the proposed technique can be applied to other process modelling notations and/or domains that benefit from this type of data-aware analysis. As additional contribution, an event log clustering technique was also developed to downsize large-event logs without compromising data relevance.

This paper is organised as follows. After introducing the overall research topic and motivation, key background concepts for helping paper understanding are presented in Sect. 2. The conformance checking proposal is introduced and explained in Sect. 3. The event-based trace clustering for log reduction is explained in Sect. 4. The proposal implementation is explained and evaluated in Sect. 5. The behaviour of the proposed conformance checking and event log reduction techniques evaluated

using synthetic data are reported in Sect. 6. This proposal is also framed in the context of related work, as discussed in Sect. 7. Finally, the main conclusions and future work prospects are provided in Sect. 8.

2 Background

This section discusses the concepts of BPMN-based process modelling, conformance checking and cluster analysis, relevant for better understanding the multiperspective conformance checking proposal presented in this paper.

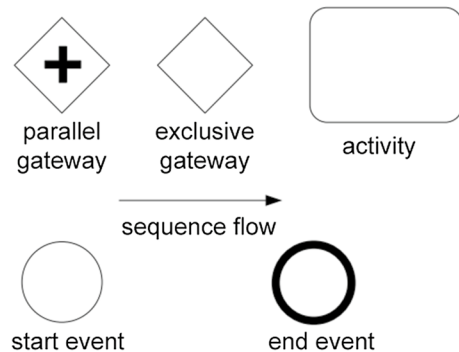
2.1 Process modelling: BPMN versus BPMN-E²

To model a business process, with process mining tasks in mind, one should consider which notation to use (*representational bias*) ensuring that process mining techniques can be executed flawlessly, without compromising the understandability of the results [4, 6]. Over the years, several process model notations were proposed, such as Petri Nets, causal nets, process trees and BPMN, being this the standard notation for business process modelling, used by a variety of professionals in their everyday jobs (business analysts, product managers, technical designers, system architects and more) [4]. For this reason, it is of great value using BPMN as the *representational bias* for process mining, at least as a starting and ending point [8], with conversions to and from more viable notations being made “under the hood”.

BPMN is a specification developed by the object management group (OMG) [9]. Its primary goal is to provide a notation that is readily understandable by all business users, from business analysts who create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and, finally, to the business people who will manage and monitor those processes.

A business process model is a network of graphical objects, which represent activities (i.e. work) and the flow controls that define their order of performance [10]. The BPMN notation provides a simple and understandable mechanism for creating business process models, being able to handle the complexity inherent to business processes. A BPMN diagram consists of a set of interconnected graphical elements, which depend on the domain and on the process being modelled. Nonetheless, there are six essential elements that together make the core of a BPMN model (see Fig. 3). These are: (a) *Start Event*, indicates where the process starts; (b) *End Event*, indicates where the process ends; (c) *Activity*, expresses the work that a company performs; (d) *Sequence flow*, shows the order of activities, i.e. how the process should flow; (e) *Parallel gateway*, indicates concurrency between activities and (f) *Exclusive gateway*, indicates exclusive decisions to be made. A list of other more specific and complex elements can be found in [9].

Fig. 3 BPMN core elements [8]



However, despite having a great support for process modelling, the BPMN notation has some drawbacks when dealing with particular domains, namely HACCP systems [7]. For these systems, two major issues were found:






1. difficulty in representing specific context details, complete workflow activities, and the semantics of the path selection to be taken during a process instance. Temporal features, identification of quality control and monitoring points and the effects of an activity on the characteristics of a product cannot be modelled, neither from a visual nor from a machine-readable perspective;
2. possible misleading conformance checking results when monitored and non-monitored activities are present in the same model.

To solve these limitations, an already mentioned extension (BPMN-E²) was devised and developed [7], introducing several new human and machine-readable elements that better represent the contextual information of HACCP processes (Table 1). Note that, despite the priority given to HACCP systems, this notation can also be easily applied to other domains with the same level of richness and expressiveness.

Considering these new elements, and the fact that they convey not only a graphical but also a machine-readable representation, richer context-aware process models can be designed carrying information about the activities being performed and the overall process.

Consider for example, the process modelled in Fig. 1 and its natural language documentation in Fig. 2, it can be modelled using BPMN-E² providing a centralised source of information about every activity of the process (see Fig. 4). Consider activity A26; previously, using the BPMN notation, it was only possible to perceive that one should “**Add medications**”; now, taking advantage of the BPMN-E² notation’s *activity effect* and *activity duration* elements, it is clear that one should “**Add medications during 60 s, adding ranitidine** to the solution’s composition, **decreasing the ranitidine’s volume in 0.1** and **increasing the solution’s volume in 0.1**”. Furthermore, considering activity A29, it is now possible, thanks to the *monitoring point* element, to know that one should “**Take the solution out from the refrigerator and check its temperature**”.

Table 1 New elements introduced by the BPMN-E² notation [7]

Elements	Definition	Graphics
Monitoring point	Represents the measurement of a variable or a set of variables in a specific point in the workflow	
Activity effect	Represents the activity effect, i.e. how the activity affects a product or how it can change the product characteristics	
Activity duration	Represents an estimation of the expected execution time of an activity	
Advanced decision point	Represents a decision point that allows to make clear the reasons involved in a particular choice, connecting the possible choices and paths with the characteristics of the product and the measured variables. It also distinguishes between Normal (above) and Quality (below) decision points	
		

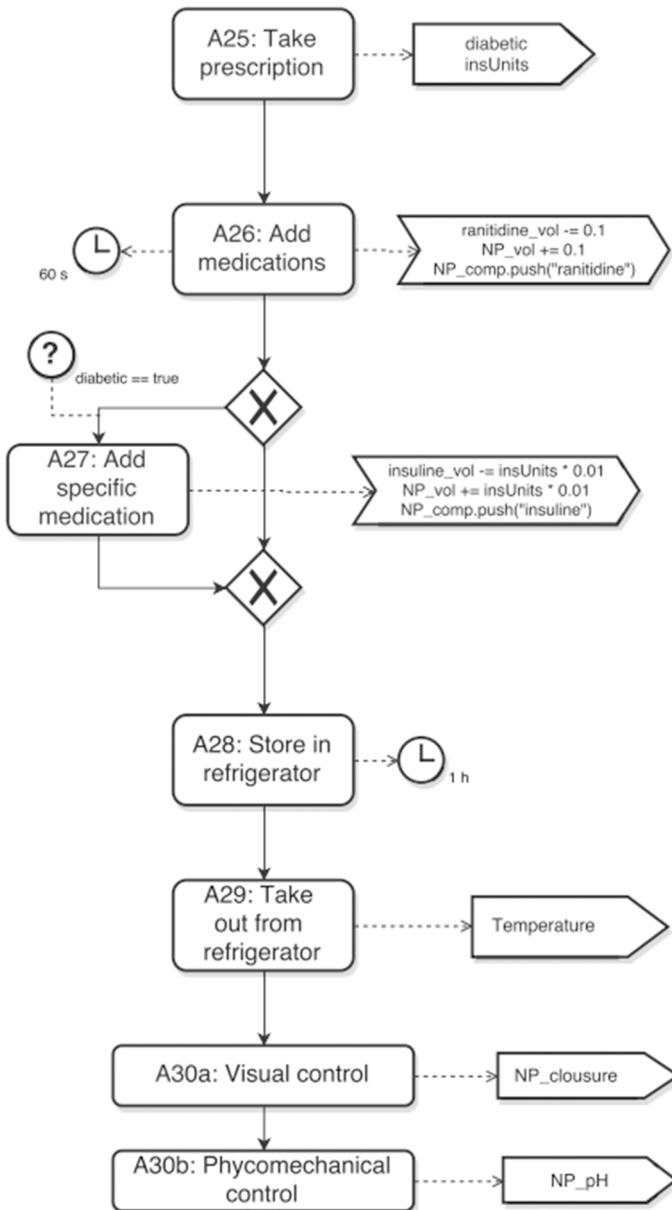


Fig. 4 Extending the model in Fig. 1 using BPMN-E² notation

2.2 Conformance checking

Most of the research in Process Mining has mainly focused on discovery techniques. In fact, looking at the literature of the last 20 years, conformance checking has

received less attention than process discovery. Models must accurately represent the reality of a process to provide trustworthy and effective decisions, however that is not always the case, with models being wrongly constructed or becoming obsolete due to changes in the business processes [11]. Conformance checking aims to pinpoint these deviations, enabling either the correction of the model or the identification of process errors. To measure the level of conformance between a model and an event log, one should be able to quantify how well the model conforms with an actual execution and vice versa. For this purpose, there are four quality criteria that together can be used to assess the quality of a given model [11–14]: (1) Fitness—the model should reflect the behaviour observed in the event log; (2) Precision—the model should not allow for a behaviour completely unrelated to the event log; (3) Generalisation—the model should generalise the example behaviour expressed in the event log; (4) Simplicity—the model should be as simple as possible (Occam’s razor). From these criteria, *fitness* is the most relevant and used metric in the literature for assessing conformance checking as it measures the proportion of the event’s log valid behaviour according to the model [15]. Therefore, most of the existing conformance checking techniques provide ways to evaluate this fitness degree. In particular, precision is a complementary metric which can be addressed in a later phase of the research; however, fitness, will be adequate in the proof-of-concept, for its primacy. Generalisation and simplicity were taken as design goals of the proposal. The value of these metrics in process discovery is discussed in [14].

2.3 Cluster analysis

Clustering is a technique that involves sorting cases or variables according to their similarity on one or more dimensions, and producing groups that maximise within-group similarity and minimise between-group similarity [16]. Clustering has been used in several fields including bioinformatics, industrial engineering, marketing, e-commerce and others [17], as an exploratory tool to help researchers and organisations to handle large amounts of data.

Clustering methods can be arranged in two different categories: *hierarchical* and *partitional*, each being composed of several individual algorithms with its own strengths and drawbacks. In this work, clustering algorithms are used to implement the *event-based trace clustering for log reduction* technique presented in Sect. 4. In particular, *K-Means* and *Density-based Spatial Clustering of Applications with Noise* (DBSCAN) algorithms are highlighted here due to their use in this work. Nevertheless, the proposed log reduction technique can be extended to allow other clustering algorithms.

2.3.1 Distance measures

To group multidimensional data, one must be able to quantify the similarities between each data point. *Distance* or *similarity* measures are therefore fundamental components in clustering algorithms [18].

The most used *distance measure* is the Euclidean distance, a special case of the Minkowski metric [18] (where $\alpha = 2$) defined as:

$$d^\alpha(z_u, z_w) = \left(\sum_{j=1}^{N_d} (z_{u,j} - z_{w,j})^\alpha \right)^{1/\alpha} = \|z_u - z_w\|^\alpha \quad (1)$$

When $\alpha = 1$, the measure is referred to as the Manhattan distance [18]. Both the Euclidean and the Manhattan measures are appealing [19]; however, using them to cluster high-dimensional can prove ineffective as the distance between the patterns increases with dimensionality. On the other hand, the cosine distance (or vector dot product)—sum of the product of each component from two vectors—is suitable for high-dimensional data [18]. There are several more distance measures that can be used for different types of data. Interested readers may refer to [19] for additional details.

2.3.2 Hierarchical clustering

Hierarchical clustering techniques are used to reveal nested structures of clusters within the data [16]. These techniques generate a cluster tree by splitting clusters into smaller ones (*divisive*) or merging clusters into larger ones (*agglomerative*) [16, 18]. Hierarchical clustering requires the selection of a distance metric, which is a unit of measurement for expressing the distances between cases, and a way of defining the link between clusters [16]. The advantages of *hierarchical clustering* are: the number of clusters does not need to be specified *a priori*, and its independence from the starting conditions [18]. However, these algorithms are computationally expensive (time and space) due to the usage and manipulation of the underlying cluster tree (or *dendrogram*). This usually makes *hierarchical clustering* inapt for larger datasets.

2.3.3 Non-hierarchical clustering

Non-hierarchical or partitional clustering, on the other hand, produces discrete clusters by dividing the dataset into a specified number of clusters [16, 18]. These techniques often use an iterative algorithm that converge to an optimal value, trying to minimise a certain criteria function locally (over a cluster) or globally (over the entire dataset) [18, 20]. Partitional clustering solves the disadvantages of hierarchical clustering; however, it requires a predetermined number of clusters that may largely impact on the final results.

K-means, introduced in [21], is the most used *partitional clustering algorithm*. The aim of *K-means* is to minimise intracluster distance [18]. With this method, initial cluster centroids (values representing the average of each cluster on each variable) are manually or randomly assigned. The algorithm then assigns cases to the cluster whose centre is the nearest based on the *Euclidean distance* between them. This assignment usually changes the cluster centroids, and thus, objects are reassigned to clusters and

the centroids are updated again. This process continues until no objects change their cluster memberships [16].

3 Multiperspective conformance checking applied to BPMN-E²

This section describes and explains the conformance checking proposal advocated in this work, starting by highlighting its main design goals, detailing then the rational and mechanisms sustaining the proposal. The present proposal extends considerably the one initially introduced in [22].

3.1 Design goals

Attending that BPMN-E² focuses on data-flows of processes, the developed conformance checking mechanism must primarily be focused on the data perspective of process mining in order to provide an effective way to pinpoint and warn against the following deviations:

- **Inconsistent activity effect** Assuming the existence of the “Activity effect” element, it should be possible to verify if one or more data variables are properly affected by an activity.
- **Inconsistent activity duration** Assuming the existence of the “Activity duration” element, it should be possible to compare the observed activity duration with the expected duration based on event timestamps. It is assumed that each event timestamp refers to the start of an activity.
- **Wrong path selection** Assuming the existence of the “Advanced decision point” element, it should be possible to verify if a specific case took the right path according to the values of one or more variables. Moreover, it should be possible to distinguish between “quality” and “normal” decision points, thus enabling taking different measures regarding which type of decision point was broken.

Furthermore, the mechanism should be able to distinguish between monitored and non-monitored activities and act accordingly. Non-monitored activities do not produce event log records; therefore, it is important to disregard these activities during conformance checking in order to reduce false negatives and provide more consistent results.

To achieve this, an initial solution was devised considering two phases: (1) *Conversion Phase*, where a BPMN-E² notation is converted to a more suitable structure; and (2) *Conformance Checking Phase*, a replay through the event log while checking with the previous structure for eventual non-conformities.

3.2 Conversion phase

To abstract the conformance checking algorithm from the initial model, there is the need for an intermediate representation of the rules to be followed when replaying the log.

One viable structure that can accurately store this information is a *Directly Follows Model* where the nodes represent the modelled activities and the edges represent a sequence flow between activities. Moreover, each edge can be annotated with the set of rules that must be satisfied when moving from one activity to another. In this way, the solution is not compromised to a specific notation or context, providing only that a conversion mechanism is available.

3.3 Directly follows models

In [23], *Directly follows models* (DFMs) are syntactically described as directed graphs in which the nodes are an activity, a start or an end. Therefore, the language of DFM consists of all traces that can be obtained when flowing from the start node to the end node.

Definition 1 (*Directly follows model—syntax*) Given an alphabet Σ such that $start \notin \Sigma$ and $end \notin \Sigma$, a directly follows rules model is a *directed graph* (N, E) , such that $N : \Sigma \cup \{start, end\}$ is a set of nodes and $E : N \times N$ is a set of edges.

Considering this, it is possible to extend a DFM by annotating each sedge with a set of conformance rules that must be followed when flowing from node n_1 to node n_2 . Consequently, this approach is here defined as *Directly Follows Rules Model* (DFRM).

Definition 2 (*Directly follows rules model—Syntax*) Given an alphabet Σ such that $start \notin \Sigma$ and $end \notin \Sigma$, a directly follows rules model is a *directed graph* (N, E, R, A) , such that $N : \Sigma \cup \{start, end\}$ is a set of nodes, $E : N \times N$ is a set of edges, R is a set of rules and $A : E \times R$ is a set of associations between edges and rules.

These rules can then be checked during conformance checking tasks to detect possible non-conformity and deviations during process execution.

3.4 SMT Solvers as conformance rules

Satisfiability modulo theories (SMT) addresses the problem of deciding the satisfiability of a first-order formula with respect to some background theory [24, 25]. An SMT Solver is a tool for deciding the satisfiability of formulas in these theories [26].

Attending to its logical nature, the problem of verifying a conformance rule can be seen as an SMT problem considering, in this case, the following background theories: *Arithmetic*, *Real* and *Arrays*. Consequently, a conformance rule is an SMT Solver instantiated with an initial set of *assertions*. In this way, the conditions of BPMN-E² elements can be represented as an assertion and *pushed* to an SMT Solver. Later, to verify conformity, for instance, temperature values recorded in an event log are also pushed into the SMT Solver as equality assertions and their satisfiability is checked. In case of satisfiability, the rule is also satisfied, conversely, in

Table 2 Conformance rules types for the corresponding BPMN-E² element

Elements	Types
Activity effect	Effect
Activity duration	Duration
Advanced decision point	Normal decision, quality decision

case of unsatisfiability, the rule is not verified. It is also important to store the type of the conformance rule in order to provide a more detailed conformance checking report. Therefore, for each new BPMN-E² element that generates a conformance rule, a corresponding type was assigned (see Table 2).

The process of verifying a rule is illustrated in the code snippet below (written in Python using Z3 solver). The SMT Solver is initialised with an assertion (`temperature > 7`), representing the conformance rule to check. Then, an equality assertion is added (`temperature == 9`), representing the actual value provided by the event log. Finally, the rule is checked. In this case, as the provided `temperature` value is bigger than 7, the conformance rule is satisfied.

```

temperature = Int('temperature')

s = Solver()

s.add(temperature > 7) # Conformance rule

s.add(temperature == 9) # Event log's temperature value

s.check() # satisfied

```

3.5 Dealing with non-monitored activities

A major advantage of BPMN-E² notation is the identification and distinction of monitored and non-monitored activities, thus providing a way to graphically represent processes with partially monitored activities without influencing conformance checking results. According to this extension, described in [7], it is possible to identify the non-monitored activities in two ways: i) within the XML, where the attributes and elements associated with this activity can be consulted; and ii) in the graphical description, where the monitored activities are connected to a Monitoring Point element.

Consider, for instance, the non-monitored activity A27 modelled in Fig. 4; traditional conformance checking approaches would be expecting records of this activity execution in the event log, which would lead to the expected yet incorrect identification of a conformance error. To overcome this drawback, non-monitored activities must be filtered during the conversion phase assuring that the inputs of non-monitored activities become connected to the corresponding outputs. The automation of

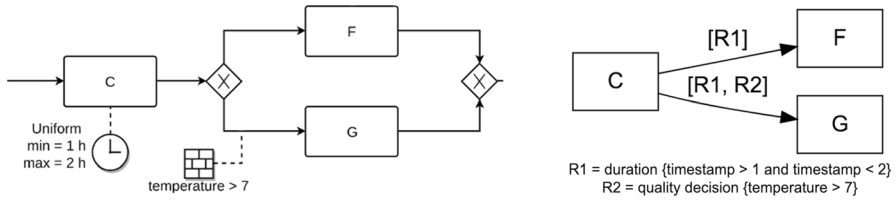


Fig. 5 Excerpt of a BPMN-E² model, extended with advanced decision points and activity durations (left), and the respective DFRM graph after conversion (right)

this process allows to maintain the consistency of the diagram and to prevent the loss of information in the workflow, and both of these problems were addressed in [7].

3.6 Putting it all together

With DFRM and SMT Solvers as conformance rules in mind, the conversion phase consists of parsing the BPMN-E² source file whilst converting the new elements into SMT Solver's assertions and associating them with the respective edge of the DFRM. An example of a conversion process input and output is illustrated in Fig. 5. In this example, activity *C* is annotated with an *activity duration* element and sequence flow (*C*, *G*) is annotated with a *quality control point* element. These elements will result in conformance rules of type *duration* and *quality decision*, respectively.

3.7 Conformance checking phase

The second phase consists of performing the conformance checking algorithm. This algorithm will receive an event log and a previously generated DFRM as inputs, and will produce a detailed report concerning the data-flow of the process as output. The event log is parsed case-by-case, activity-by-activity keeping record of the *current* and *last* activities being replayed. Considering the attributes of both activities, it is possible to extract: (1) the activity's duration; (2) the activity's effect; and (3) the path followed. The pseudo-code for this procedure can be:

```

For each case in log:
    last_event = None
    For each cur_event in case:
        verify_rules(dfrm, last_event, cur_event)
        last_event = cur_event

```

During this phase, all detected non-conformities are recorded, storing the conformance rules that were broken, the cases they occurred in, and the corresponding activities and parameters. In this way, it is possible to produce a report regarding [7]:

- *Time*—stating the correspondence between the theoretical time constraints and the real time taken by the activities (measured through the event log timestamps).
- *Activity effects*—stating the expected activity effect over a process instance with the real effect carried out.
- *Quality points*—stating the fulfilment of checkpoints on the workflow (checking whether an instance has followed the correct path).

Alongside these reports, it is also possible to detect control-flow deviations when the sequence flow observed in the event log does not exist in the DFRM. However, we recognise that there are better and more complete methods to tackle control-flow conformance checking (namely *alignments*). This is why, we propose a combination of state-of-the-art *alignment-based* methods with our approach to provide a richer analysis regarding both control and data-flow perspectives.

Considering the four quality criteria (*fitness*, *precision*, *generalisation* and *simplicity*) that can be used to access the quality of a given model [1, 11, 12, 14], we propose a way to compute the *fitness* of a model, as it measures the proportion of the event's log valid behaviour according to the model [1]. Although there are several ways to calculate a quantitative measure, regarding the evaluation of fitness, we have adopted well-established formulas proposed in the process mining literature, which confers confidence in the obtained results. In this way, for an event log l and a model m , the *fitness* values for each individual conformance rules of type t (*effect*, *duration*, *normal decision*, *quality decision*) are computed using Eq. (2):

$$f(l, m, t) = \begin{cases} \frac{\text{satisfied}_t}{\text{total}_t}, & \text{if } \text{total}_t > 0 \\ 1 & \text{if } \text{total}_t = 0 \end{cases} \quad (2)$$

where satisfied_t is the number of conformance rules of type t that were satisfied and total_t is the total number of conformance rules of type t that were tested. Then, the overall fitness value (f_{cr}) is computed as a weighted average of fitness values for each type of conformance rules, calculated previously (see Eq. (3)). In this way, it is possible to assign higher relevance to certain types of deviations, e.g. *activity effects* errors can be more critical than *activity duration* errors resulting in a heavier weight factor.

$$f_{cr} = cr_fitness(l, m) = \sum_{t \in \text{types}} w_t \cdot f(l, m, t) \quad (3)$$

In Eq. 3, *types* is the set of types of conformance rules and w_t their corresponding weights. Note that the sum of the weights must be equal to one. Finally, we can combine both control-flow alignment-based fitness value with the fitness value of our approach, according to Eq. (4):

$$fitness(l, m) = w_t \cdot f_t + w_{cr} \cdot f_{cr} \quad (4)$$

where w_t and f_t are the weight and the fitness values of a state-of-the-art “traditional” conformance checking approach, and w_{cr} and f_{cr} are the weight and the fitness values of our proposal.

4 Event-based trace clustering for log reduction

This section introduces the event-based trace clustering technique proposed in this work to downsize large-event logs.

4.1 Motivation

One of the main drawbacks of multiperspective conformance checking techniques compared to more common control-flow approaches is that considerably larger amounts of traces must be analysed in run-time, which unavoidably leads to longer execution times. As an example, let’s assume an event log with a thousand different cases that span ten trace variants; control-flow approaches only need to compute the fitness for these ten variants and map it to the according cases. On the other hand, however, adding the data perspective leads, in the worst case scenario, to a number of trace variants equal to the total number of cases. Therefore, data-flow approaches often need to compute the fitness for every single case in the event log.

With this in mind, and acknowledging that there is no “workaround” to this reality at the algorithm level (the conformance checking task will always compute fitness for all individual event log cases), a way to downsize the event log while retaining the most relevant cases (data wise) is proposed. In fact, this can be accomplished by sampling cases with distinct data-flow behaviour and excluding those with similar behaviour.

In this scenario, attending to its importance in pattern recognition [18], cluster analysis can be leveraged to group cases based on the values of different event attributes and their changes during process execution.

4.2 Vectorising a trace

A clustering algorithm operates over a number of data points called *feature vectors* [18]. In turn, a *feature vector* is composed of several *features* (or *attributes*). To be able to execute clustering algorithms over event log traces, these must be transformed into a corresponding *feature vector*, with its *features* being the event’s attributes.

A vectorised trace can then be defined as an object whose items are the attributes of all events in a log and their values are the corresponding attributes’ data for that particular trace. In order to vectorise a trace, the following steps are taken:

1. Assess which attributes should be present in the *feature vector*. Optionally, *timestamps* can be leveraged to compute an activity duration, thus taking time perspective into consideration during clustering.
2. Create a *feature vector* for each trace's event containing the chosen attributes' values. If a particular event does not contain one or more of the selected attributes, they can simply be omitted.
3. Append all the previous generated event *feature vectors* to form the final *feature vector*.

The process of trace vectorisation is illustrated in Tables 3 and 4.

4.3 Clustering and sampling

The similarity between the different *feature vectors* is computed based on a *distance measure*, used to evaluate how close two *vectors* are to each other. There are several *distance measure functions* that can be used to assign a level of similarity between *data points*, such as *Euclidean distance*, *city block distance*, *Minkowski distance*, *Canberra distance*, *cosine distance*, among others [19]. Here, the *Euclidean distance* will be used (defined in Eq. 1), as it is the most popular distance measure; however, one should keep in mind that other approaches can yield better results depending on the underlying problem.

We will not delve into the inner works of clustering algorithms. Interested reader are referred to [20]. Instead, we assume their output as a group of clusters, each one representing a *hard partition* of the original data with close proximity in regard to a number of *features*. The underlying motif of using clustering analysis to perform event log reduction lies in considering that all the traces belonging to a cluster provide the same amount of process knowledge as its neighbours and, therefore, can be removed from the original event log while keeping an accurate representation of how the process was conducted—this is hereby defined as *trace sampling*. A high-level view of this process is represented in Fig. 6.

It is natural having clusters with different sizes, each containing a different number of cases. To assure that the distribution of event behaviour reduced event log remains unaltered from the original log, the amount of sampled cases must be proportional to the cluster's size. In this way, the reduced event log keeps, not only the original's data-flow behaviour, but also the proportion in which this behaviour occurs. Take, as an example, an event log with 10 different cases, where 8 of them follow a specific data-flow and the remaining 2 follow a different data-flow. Consider now that a 50% reduction is necessary; it is important that the final event log maintains this 8:2 ratio, by sampling 4 cases from the first cluster and 1 case from the second cluster.

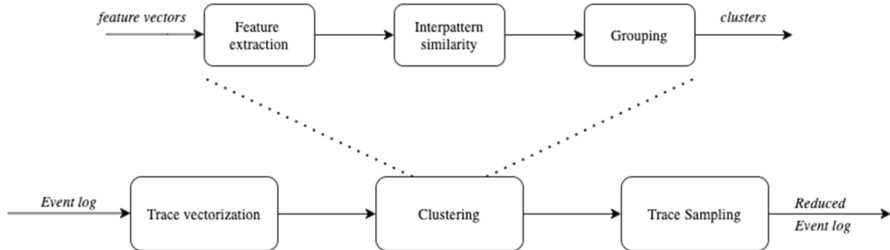
The main downside of event log reduction is the eventual loss of process information due to the exclusion of several traces from the original event log. Even though the most representative cases are kept in the reduced version, can oftentimes exist traces that behave “oddly” compared to the rest, the so-called *outliers*. These *outliers*

Table 3 Excerpt of the event log used in Sect. 5. Used here to illustrate the vectorisation of event log traces

Case:concept:name	Concept:name	Time:timestamp	Amount	Expense	PaymentAmount	TotalPaymentA- mount	Dismissal
A10005	Create fine	2007-03-20 00:00:00+01:00	36.0	-	-	0.0	NIL
A10005	Payment	2007-03-21 00:00:00+01:00	-	-	36.0	36.0	-
N28738	Create fine	2000-09-13 00:00:00+02:00	31.0	-	-	0.0	NIL
N28738	Send fine	2000-10-30 00:00:00+01:00	-	6.71	-	-	-
N28738	Payment	2000-12-05 00:00:00+01:00	-	-	38.01	38.01	-

Table 4 Vectorised traces extracted from Table 3, using the attributes *amount*, *dismissal*, *paymentAmount* and *expense*

Case	Create fine			Send fine		Payment
	Amount	PaymentAmount	Dismissal	Expense	PaymentAmount	PaymentAmount
A10005	36.0	0.0	0 (NIL)	–	–	36.0
N28738	31.0	0.0	0 (NIL)	6.71	0	38.01

**Fig. 6** Event-based trace clustering for log reduction: stages. Clustering stages follow [20]

cause two drawbacks: a) they may be assigned to clusters that do not reflect their actual behaviour and end up being excluded; b) *outliers* are known to negatively impact certain clustering algorithms disabling them from correctly identify clusters [27]. With this in mind, one should evaluate the process data available and judge if the benefits of having a reduced event log pay off for the loss of trace information. As a rule of thumb, log reduction can prove useful when individual traces can be neglected by generalisation, such as discovering a process model from an event log or computing conformance checking fitness and precision. If the context calls for a mandatory analysis on each trace to pinpoint deviations and its causes, a reduced event log would naturally limit the analysis by omitting several cases. Nevertheless, even in those cases, the reduced version can be used as a starting point to provide a general picture of the problem in hands, that could later be extended if a more thorough analysis is needed.

This technique was mainly developed as a solution to the long execution times experienced during the previously proposed conformance checking technique; however, as it operates at the log level, one can use it to downsize any event log and apply it to other data-aware Process Mining techniques, namely conformance checking, process discovery and process enhancement.

4.4 Summary

In this session, the event-based trace clustering for log reduction algorithm was explained. Framed in the context of the present work and inspired by trace clustering

techniques, its main goal was the transformation of existing event logs by sampling the cases that better generalise the business process. The proposed technique relies on three stages: *trace vectorisation*—pre-processes the event log, based on user-selected criteria, to feed state-of-the-art clustering algorithms; *clustering*—groups the previously created trace vectors into heterogeneous clusters; and *sampling*—selects a percentage of cases from each of the computed clusters. It is expected that each cluster contains cases that behave similarly in regard to the selected attributes and, therefore, can be considered redundant for certain process mining tasks.

Although it was developed in the context of the data-aware conformance checking algorithm proposed in Sect. 3, this technique was designed to work independent of any Process Mining technique and therefore can, and should be, leveraged as a tool for event log processing.

5 Proof of concept

The main goal of the developed conformance checking task is to provide detailed insights into possible inconsistencies found regarding three main aspects: (a) activity effects; (b) activity duration; and (c) path selection. To achieve this, the results of verifying all conformance rules are recorded alongside contextual data regarding the moment the rule was verified, including the case, the activities and the event log values being processed. These data can then be processed in multiple ways to meet the analytical goals of the users.

The whole approach is automated resorting to the implementation of a Python library providing the following features: [Objects] (i) BPMN-E2 implementation as an extension of `bpmn-python` project¹; (ii) DFRM class implementation; (iii) `RuleParser`, used to translate a rule from its BPMN-E2 notation to a correspondent valid Z3 assertion; and (iv) `MonitoringParser`, used to translate a BPMN-E2 monitoring point to a correspondent valid Z3 assertion. This requires a different parser since monitoring points don't represent assertions *per se*; [Conversion] for allowing BPMN-E2 conversion to DFRMs; [Visualisation] for constructing BPMN-E2 diagrams; and [Conformance Checking] for providing the three different strategies of multiperspective conformance checking that can be used by the end user.

Using the developed library, an out-of-the-box HTML report similar to the one in Fig. 7 can be generated, containing several sections providing information and visualisations regarding the process model and the event log, and fitness values (in the case, rule and type level). It is also possible, and encouraged, to output the results using dataframes enabling a more customised and powerful data analysis meeting the user's business needs.

In this section, the proposed conformance checking technique is tested on a real-life scenario and evaluated using synthetic data. It starts by providing a real-life use-case analysis and showing how the developed library can be leveraged to perform

¹ Project for creating a Python library that allows to import/export BPMN diagram (as an XML file) and provides simple visualisation capabilities, available at <https://github.com/KrzyHonk/bpmn-python>.



Fig. 7 Example of a conformance checking HTML report

multiperspective conformance analysis. In Sect. 6, time complexity and performance are assessed using synthetically generated data. The event log reduction technique introduced in Sect. 4 is also evaluated in this section.

5.1 Real-life use case

In the proof-of-concept, a real-life event log taken from an information system of the Italian police [28] was considered, in order to increase the meaningfulness of the study and corresponding results.

5.2 Event log

As stated in [28], the event log respects to a road traffic fine management process regarding the creation, payment and appeal of fines. The event log is rich in attributes that can be leveraged to support a data-aware conformance analysis. After a thorough analysis, we verify that the log is composed of 561.471 events recorded across 145,800 cases, which were recorded between January 2000 and June 2013.

From Table 5, the following attributes can be identified: (i) *totalPaymentAmount*—total amount already paid by the offender; (ii) *paymentAmount*—amount paid by the offender during the “Payment” event; (iii) *amount*—monetary value of the fine; (iv) *dismissal*—flag indicating possible cause for fine dismissal. “NIL” if the fine is not dismissed, “G” if it is dismissed by the prefecture, and “#” if it is dismissed by a judge; (v) *expenses*—monetary value of extraordinary expenses such as the cost of sending the fine via mail.

5.3 BPMN-E2 process model

As mentioned in Sect. 2, BPMN-E2 is an extension of BPMN that has been proposed as a way to add context information to the BPMN model. The details of this extension are described in [7]. In the current work, we have followed the following order: (i) the petri net described in [26] was taken as background, (ii) the behaviour of the process was captured in an equivalent BPMN, (iii) the model was increased using the BPMN-E2 extension in order to contemplate the inclusion of context information, iv) the evaluation tests were carried out on the final BPMN-E2 model.

In more detail, the business process model was designed considering the Petri net illustrated in [28], being the general process flow as follows. When a road traffic offense occurs, a fine is created. The offender can then pay the fine partially or in full at several stages of the process. The fine management is closed when the offender pays the full amount. If the process is still open, a fine notification will be sent to the offender's residency, after which he can choose to appeal of the decision (to a judge and/or to the prefecture). If the fine is not paid before 180 days, a penalty is added to the current amount of the fine. If the fine is still not paid, it will eventually end by handing over the case for credit collection.

This process can then be enhanced with several BPMN-E² elements to provide a data-flow perspective at the activity level (see Table 6), resulting in the final BPMN-E² model, as shown in Fig. 8.

In addition, BPMN-E² formally specifies that is mandatory to define a *Monitoring Point* for all monitored activities. Therefore, all activities were extended with a *Monitoring point*. However, most of these elements were left out from Fig. 8 for comprehensibility purposes. The resulting diagram constitutes a centralised and easy-to-understand source of knowledge related to the process being analysed.

5.4 Analysing the results

Initially, the conformance checking task was used to resolve an overall fitness value to access how well the event log fit the process model. On average, 3.68 conformance rules were tested per case, reaching a total of 554.291 tested rules. Each case took, on average, 1.29ms to be processed. A fitness value of 0.813 was reached assuming the same weight for every conformance rule. However, it was decided that the process *end condition*—a process should only end when dismissed or when the fine is fully paid—had to weight more than the others given the severity of its possible violation. Therefore, considering a 5 times increase in this conformance rule weight, a new fitness value of 0.81 was reached. The minor change in fitness can be understood as a sign that this rule was not broken that many times. In fact, there were only 6986 (4.6%) deviations recorded for this particular conformance rule. Despite the low number of occurrences, it still added up to 329.069,54 EUR in financial injury. It is also relevant to note that the activity “Add Penalty” failed to increase the fine's amount 13 times, although not much, these deviations may lead to additional financial losses.

Table 5 Excerpt of the event log from the road traffic fine management process

Case:concept:name	Concept:name	Time:timestamp	Amount	Expense	PaymentAmount	TotalPay-mentAmount	Dismissal
A10004	Create fine	2007-03-20 00:00:00+01:00	36.0	-	-	0.0	NIL
A10004	Send Fine	2007-07-17 00:00:00+02:00	-	13.0	-	-	-
A10004	Insert Fine Notification	2007-07-24 00:00:00+02:00	-	-	-	-	-
A10004	Add penalty	2007-09-22 00:00:00+02:00	74.0	-	-	-	-
A10004	Send for Credit Collection	2009-03-30 00:00:00+02:00	-	-	-	-	-
A10005	Create Fine	2007-03-20 00:00:00+01:00	36.0	-	-	0.0	NIL
A10005	Payment	2007-03-21 00:00:00+01:00	-	-	36.0	36.0	-
A16149	Create Fine	2007-10-24 00:00:00+02:00	36.0	-	-	0.0	NIL
A16149	Send Fine	2008-01-18 00:00:00+01:00	-	13.0	-	-	-
A16149	Insert Fine Notification	2008-01-31 00:00:00+01:00	-	-	-	-	-
A16149	Insert Date Appeal to Prefecture	2008-02-12 00:00:00+01:00	-	-	-	-	-
A16149	Add penalty	2008-03-31 00:00:00+02:00	74.0	-	-	-	-
A16149	Send Appeal to Prefecture	2008-03-31 00:00:00+02:00	-	-	-	-	#

Table 6 Rules extracted from the description of the process

Activity	Rule type	Description
Send fine	<i>Duration</i>	Fine notification is due to be sent within 60 days after fine creation
Appeal to Judge	<i>Duration</i>	Offender has 60 days to appeal to judge after being notified
Add Penalty	<i>Effect</i>	When a penalty is added the fine's payment amount must increase
Any → Insert for credit collection	<i>Decision</i>	The fine goes to credit collection; it implies that the offender did not pay the fine's full amount
Send Appeal to Prefecture → end	<i>Decision</i>	If the appeal to prefecture succeeds, the fine is dismissed with flag "G"
Appeal to Judge → end	<i>Decision</i>	If the appeal to the judge succeeds, the fine is dismissed with flag "#"
Any → end	<i>Decision</i>	The process should only end when the fine is dismissed or fully paid

Shifting the perspective to the rule level, it was possible to compute a fitness value for each conformance rule, as shown in Fig. 9. This allows to conclude that most of the recorded deviations are directly related to wrongly set dismissal values and delays in sending the notifications of fines. More precisely, 38% of times the fine was dismissed by the prefecture with a wrong dismissal value, in 39% the fine was dismissed by the judge with a wrong dismissal value and in 48% the fine was sent after the fixed limit to notify fines.

The analysis that follows was driven by a set of questions, to which an answer was sought. They are:

5.4.1 What is the average number of deviations per case?

The average number of deviations per case is relatively low, set at 0.39 *deviations per case*, as shown in in Fig. 10 (small triangle). More precisely, there are 54.812 cases with at least one deviation, corresponding to 36.5% of the total. Analogously, the remaining cases have no data-flow deviations (73.5%). The maximum number of deviations recorded in one individual case was 3; however, only 10 cases reflected this behaviour. These conclusions were supported by Fig. 10.

5.4.2 What is the percentage of deviations per type of conformance rule?

From the detected conformance rule deviations, it was possible to identify that all the rules were at least broken once. Nevertheless, a clear amount of deviations was directly related to activity duration; in fact, 86% of deviations were of this type. 14% were of type "*decision*" and less than 1% were of type "*effect*". This shows that activity effects are taking place in a correct manner whilst activity duration is being violated more often and should be targeted for a more detailed analysis. These conclusions were supported by Fig. 11.

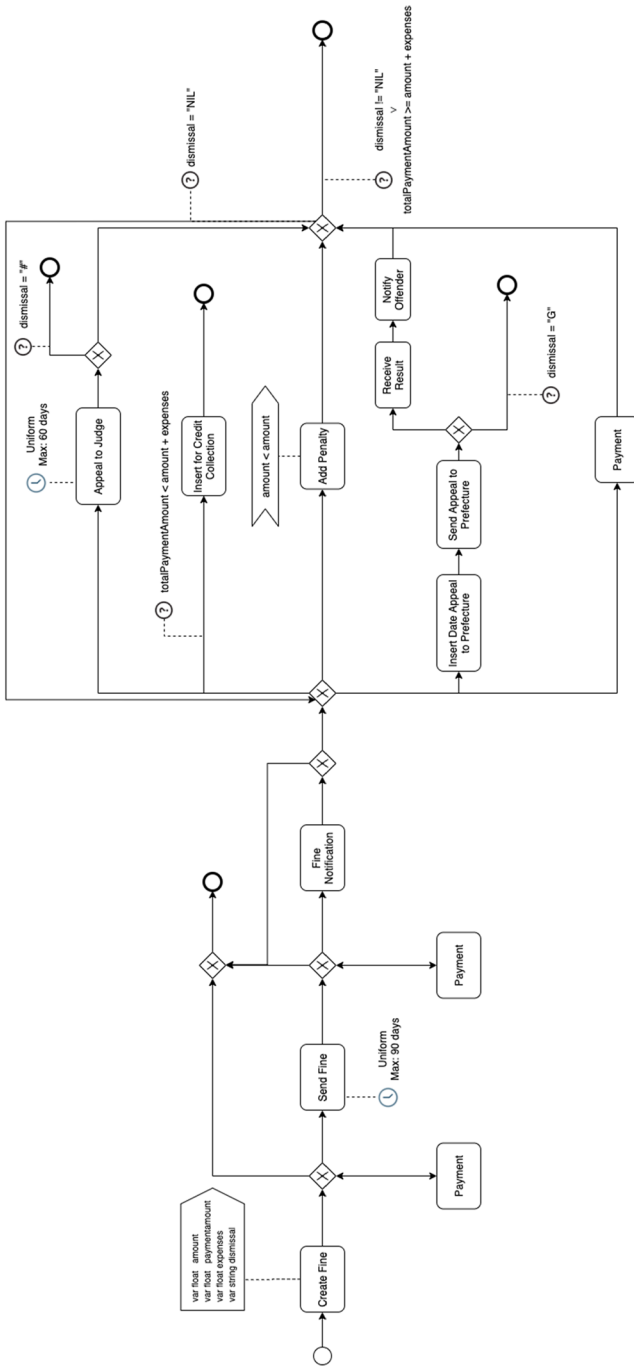


Fig. 8 BPMN-E² model of the road traffic fine management process

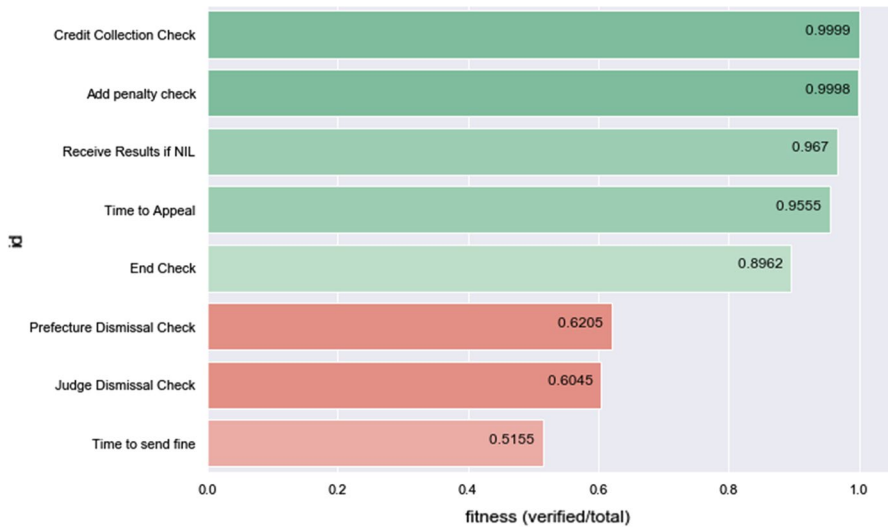


Fig. 9 Fitness values for each conformance rule: satisfactory results (in shaded green) vs. less than ideal results (in shaded red), for a threshold of 0.7

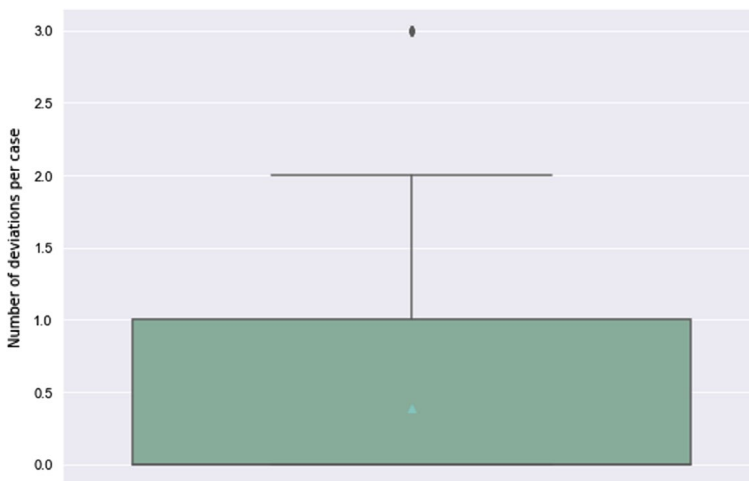


Fig. 10 Boxplot of the number of deviations per case

5.4.3 What is the average activity duration when the corresponding type of rules fail?

For this particular scenario, two rules were defined: *Time to Send fine (AD1)* and *Time to Appeal (AD2)*. After an initial analysis, it was found that *AD1* suffered, approximately, 99% of this deviation type. This points out irregular behaviour

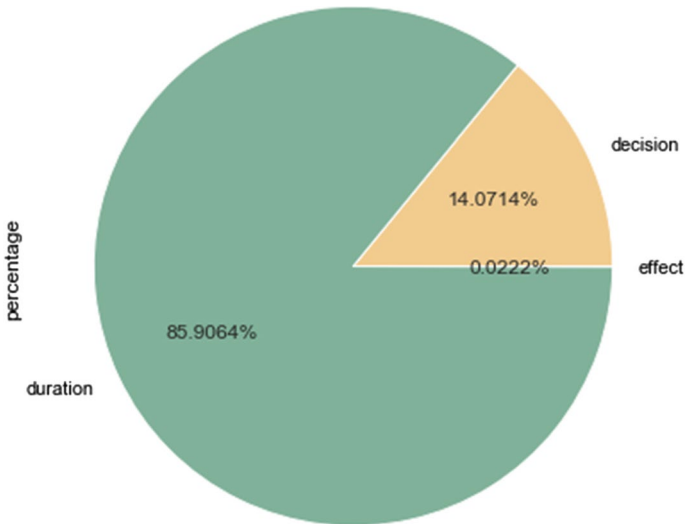


Fig. 11 Percentage of deviations per type of rule

regarding the process of mailing the fine notification to the offender. Despite the Italian law establishes a maximum of 90 days to notify a traffic fine, when this law is broken, the average notification delay reaches 123 days, i.e. one month above the defined limit.

In respect to rule *AD2*, despite being broken considerably less times, the delays were significantly larger. The maximum time to appeal to a judge is established in 60 days. However, when this limit is not satisfied, appeals take on average 279 days, i.e. more than 6 months above the defined limit. These conclusions were supported by Fig. 12.

6 Evaluation using synthetic data

Several synthetic event logs were generated for evaluating the behaviour of the proposed conformance checking and event log reduction techniques.

6.1 Experimental setup

The conformance checking experiments were mainly focused on the time performance and scalability of the proposal, specifically, when increasing the number of: (i) *conformance rules in the model*; (ii) *cases in the event log*; and (iii) *events per case in the event log*.

With this in mind, four BPMN-E² diagrams were created. All of them are composed of five activities (A, B, C, D, E) and five monitoring groups. The first diagram (referred to as activity duration (AD)) contains two extra activity duration elements; the second diagram (activity effect (AE)) contains two extra activity effect elements;

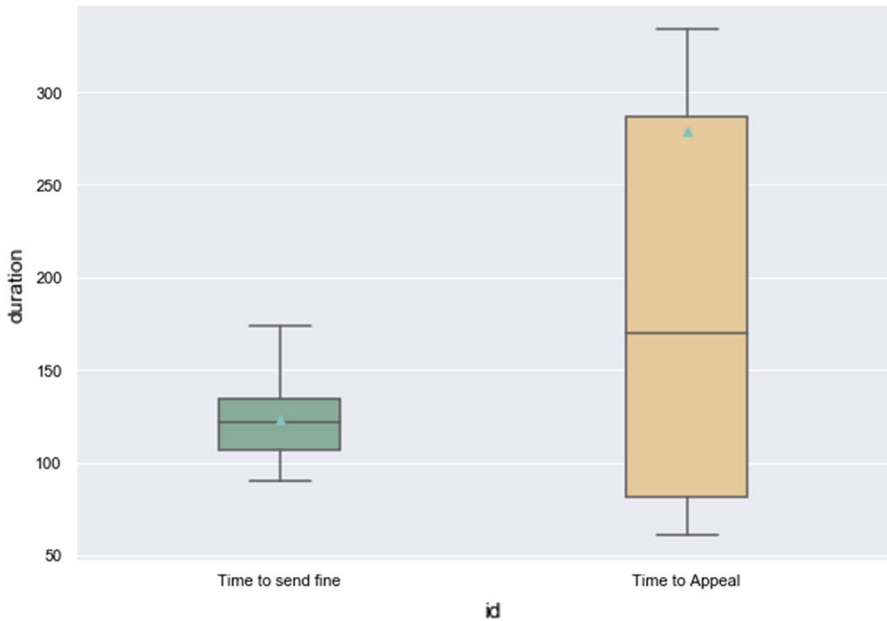


Fig. 12 Boxplot of duration for each duration-type conformance rule

the third diagram (decision point (DP)) contains two extra decision point elements; and finally, the fourth diagram (ALL) contains all three elements from the previous diagrams. Figure 13 illustrates this last diagram (ALL); the others follow exactly the same diagram but keeping only one type of extension elements.

The event logs, on the other hand, were designed to evaluate the scalability of the approach. It was assured that every event log was fully compliant regarding the control-flow perspective. In turn, each event log was generated with a combination of the following characteristics: number of cases (1k, 10k and 100k) and number of events per case (2 and 4). This results in 6 different event logs with increasing number of cases and events.

The following tests were executed in a computer with a 2,5 GHz Quad-Core Intel Core i7 CPU and 16 GB of RAM.

6.2 Evaluating conformance checking

To test the conformance checking approach, each event log was ran against each process model, for a total of 24 tests. Each test was executed five times, the test results were then averaged to obtain a final result, as shown in Table 7.

Analysing the obtained results, it is possible to assess the following in regards to the overall performance of conformance checking. By isolating the event log, it is noticeable that verifying the same amount of rules takes approximately the same time regardless of their type. This was expected as every rule is built over a similar

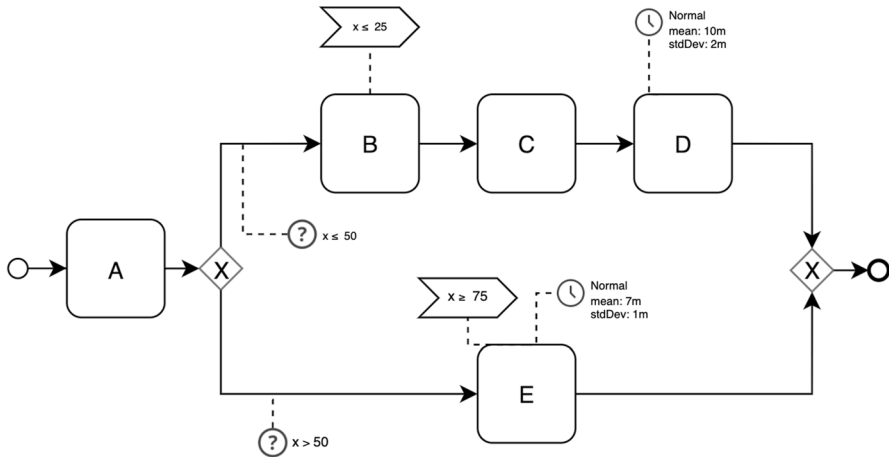


Fig. 13 Diagram ALL used during evaluation. Note that for each path, despite the distinct number of activities, the number of rules to test was kept

SMT Solver. Furthermore, the conformance checking task scales linearly regarding the number of rules being tested per case. This can be backed up by the obtained results, considering a linear increase in the task duration for bigger logs that comprise, naturally, more rules to be verified. The number of events per case, and consequently, the total number of events, proved to be irrelevant to the overall performance as the execution time is hardly affected by any change in this value, if the number of rules per case is kept constant.

Considering both the results in Table 7 and the *heatmap* in Fig. 14, it is evident that the execution time is mainly affected by the number of rules that are tested. The proposed conformance checking task runs in $\mathcal{O}(N)$, with N being the number of rules to test. This number is indirectly affected by the number of cases (\mathcal{C}) and the average number of rules per case (\mathcal{R}). In this way, the previous time complexity can be rewritten into $\mathcal{O}(\mathcal{C} * \mathcal{R})$. Thus, the solution's performance will decrease for larger event logs and more *rule-dense* BPMN-E² diagrams.

6.3 Evaluating event log reduction

The following experiments and discussion aim at assessing the performance of our proposal for event log reduction. For this, each clustering technique will be used to perform increasingly heavier log reductions while verifying how it impacts both on the overall representativeness of the original event log and on the conformance checking task. To evaluate the goodness of clustering results, *clustering validation* will be used, as it has long been recognised as one of the vital issues essential for the success of clustering applications [29]. There are two main ways to perform clustering validation: (a) external clustering validation; and (b) internal clustering validation. The former uses external criteria to validate the results, e.g. cluster labels, whilst the latter uses features inherent to the data itself

Table 7 Evaluation results

Model	Cases	Events per case	Execution time (s)	
AD	1k	2	0.3142	
		4	0.3676	
	10k	2	2.9093	
		4	3.3608	
	100k	2	28.9304	
		4	33.3462	
	AE	1k	2	0.3520
			4	0.3868
10k		2	3.4969	
		4	3.6441	
100k		2	35.0104	
		4	36.5146	
DP		1k	2	0.3372
			4	0.3668
	10k	2	3.2929	
		4	3.4433	
	100k	2	33.3938	
		4	33.6114	
	ALL	1k	2	1.0125
			4	1.0740
10k		2	9.5141	
		4	10.0135	
100k		2	95.4211	
		4	99.3699	

[19, 29]. Commonly, it is hard to find real datasets that have prior cluster information [29], making internal validation the only way to validate such problems.

In this particular scenario, only external cluster validation techniques were used, by comparing the fitness values obtained from the conformance checking task run over the original event log with the fitness values computed using its downsized versions. In this way, it can be assessed how different clustering algorithms and parameters impact on the event log by cross-referencing their fitness values with the unaltered event log.

The technique was applied to the event log introduced in Sect. 5.1. Five different percentual downsizes (50%, 40%, 30%, 20%, 10%) were applied for both K-means and DBSCAN clustering algorithms. Each downsized log was then fed into the developed conformance checking algorithm. The introduced error is computed simply by comparing the fitness value obtained using the reduced event log with the fitness value obtained from the original event log, using the absolute value of the difference between them. In this way, it is possible to measure the deviation of results when different reduction parameters are used. As this technique relies heavily on how well the data is clustered, relevant parameters were

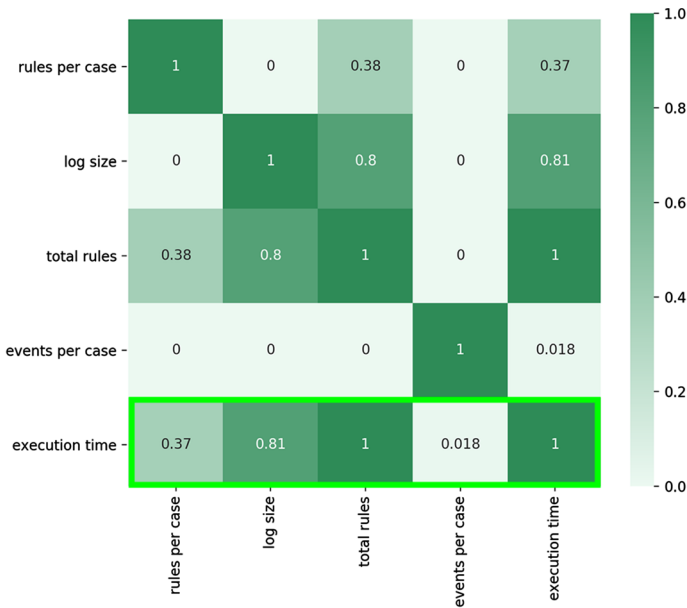


Fig. 14 Evaluation heatmap: execution time correlations are highlighted

tweaked for each clustering algorithm. Different *number of clusters* and *eps* values were used for K-means and DBSCAN, respectively.

Analysing the graphs in Fig. 15, it is noticeable that the difference in fitness value for this particular scenario is minimal. In fact, the maximum error value does not exceed 0.0006, which is greatly satisfactory. In general, the error increases for heavier downsizes with K-means performing better overall regardless of the insignificant difference between the two clustering algorithms. Nevertheless, there are cases where heavier downsizes yield better results. In fact, K-means algorithm performs at its highest on a 40% reduction with 25 clusters. On the other hand, DBSCAN algorithm excels on both 30% and 50% reductions with an *eps* value of 0.5.

Considering that time complexity of the conformance checking algorithm is $\mathcal{O} \leftarrow \mathcal{C} \uparrow \mathcal{R} \Rightarrow$, with \mathcal{C} being the total number of cases, i.e. the event log size, and \mathcal{R} the average number of rules per case, reducing the event log is bound to reduce the algorithm’s execution time. In fact, it is expected that a reduction percentage in the number of cases will result in the same reduction percentage in the execution time. For the case under consideration, it is possible to reduce the original execution time from around 3 min to 18 s without any significant change in the final result, by applying a 90% event log reduction.

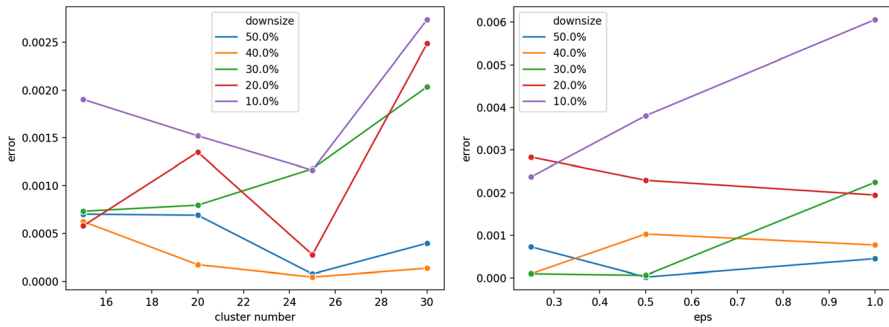


Fig. 15 Conformance checking errors after applying the event log reduction technique, using K-means (left) and DBSCAN (right) clustering algorithms

7 Related work

The lion's share of process mining research focuses on control-flow, i.e. the ordering of activities [30]. Therefore, the majority of the effort put on conformance checking techniques is pointing towards the control-flow perspective, ignoring other perspectives such as data, resources and time [31]. In this way, there can be several deviations that are not caught during a conformance checking task, e.g. activities that take longer than what is expected, activities that should not be executed by a certain resource or activities that fail to make specific changes.

Considering that focusing only on one perspective can lead to an incomplete diagnosis, data-aware (also called multiperspective) conformance checking techniques were developed [30–35]. In [32], a plugin for ProM is proposed to evaluate whether the recorded executions of a process match its corresponding model. In the process, UML and Petri nets are used as background languages to include artefact-based descriptions. In [31], a technique is proposed that extends control-flow alignments to incorporate other perspectives by constructing an *Integer Linear Programming* (ILP) problem and, consequently, solving it. In [35], a different approach is proposed using *Compliance Rules Graphs* (CRG) [36] to declare a set of rules that the process execution must obey, each rule is bound to an activity, thus enabling to pinpoint the cause of a violation. Other works, such as [33], propose new approaches to the use of conformal checking techniques in specific application contexts by combining novel Big Data techniques. Considering that a multidimensional approach to conformance checking may not be feasible to apply to real-life event logs due to long computations, in [28] an algorithm is proposed balancing the deviations with respect to all the perspectives based on a customizable cost function. This approach has been validated through empirical evaluation using a real-life event log and a process model provided by the local police of an Italian city (also used for evaluation in the present article). More recently, in [23], process mining discovery and conformance approaches using *Directly follows Models* were proposed, given its intuitiveness and simplicity; however, these techniques focused on control-flow instead of data-flow. As explained in Sect. 3, DFMs served as the baseline for this work, being extended with conformance rules to accommodate data-aware conformance checking.

To achieve this data-aware multiperspective in conformance checking techniques, it is mandatory to have the necessary data available. In the presented proposal, the chosen strategy takes into account those data reflecting domain knowledge information represented in conjunction with the process model. In this regard, the used BPMNE² extension offers the possibility to detail the workflow behaviour, the activities being performed and the context of one particular process. A literature analysis shows that there is a trend to incorporate different knowledge in the process model to improve the applicability of process management techniques. In this line, [37] emphasises the importance of understanding the application context information for enhanced process analysis. The work discussed in [38] proposes a system designed to extract general knowledge of the context using previous product cases, which may be used, for example, to manage conflicting products. In [39], a pre-warning analysis system that analyses product abnormalities from a data-centric point of view is developed. The work described in [40, 41] highlights the formal definition of the new modelling elements for the monitoring events generated by processes. These modelling elements are especially useful for other work of the same authors, about the utilisation of conformance checking techniques in hospital environments [42]. The researches [43–45] discuss the idea of further developing the description of decision points. The work described in [46] proposes five multiperspective process mining methods that deal with the interaction of multiple process perspectives. It includes the development of perspectives related with conformance checking (more aligned with the proposal described in this paper) and others focused on process discovery. Finally, other works such as [47–49] develop the concept of providing additional information related to specific activities. Nevertheless, these works analyse specific issues of data representation; meanwhile, the BPMNE² extensions offers a multipoint of view perspective of different types of information. This aspect allows the creation of multiperspective conformance checking techniques as the one proposed in this paper.

This literature trend on the incorporation of different types of information in the process model follows several ideas pointed out by the Process Mining Manifesto [6], elaborated by the IEEE Task Force on Process Mining. In this work, a collection of guiding principles and future challenges about process mining techniques are analysed. In particular, it highlights the importance of achieving better data quality and completeness to improve the applicability of current techniques. Moreover, this can foster the development of novel process mining techniques that take advantage of the new information to obtain enhanced results.

Several clustering techniques have been used in the field of Process Mining, namely to perform *trace clustering*. In [50], *agglomerative hierarchical clustering* is used to automatically identify *process execution classes* based on selected case attributes. The main *clustering unit* of their work is a *sublog* (i.e. a set of cases). They start by dividing the original event log into smaller subsets and then merge them together based on their similarity until the defined number of clusters is reached. The returned clusters form *sublogs* that can then be individually analysed. Despite considering selected case attributes to initially slice the event log, this approach relies solely on the control-flow distance to compute *sublog similarity*.

Regarding the topics of *cluster analysis* and *trace clustering*. In [51], *agglomerative hierarchical clustering*, *K-Means*, *Quality Threshold* and *Self-Organising Maps* were used to perform *trace clustering*. The concept of *trace profile* is introduced as a set of related items that define a trace from a specific *perspective*. Each trace profile can be then aggregated in a *feature vector* and fed to a cluster algorithm. Similarly to [50], this approach aims to return a set of more homogeneous *sublogs* based on the chosen *trace profiles*. The main advantage of this approach is the multiperspective nature of the *profiles*. As an example, the authors introduce: the *activity profile*—specifying which activities were executed; the *originator profile*—counting how many events have been caused by each originator per trace; the *event profile*—counting how many events are annotated with a specific attribute; amongst others. In [15], the same authors compare several dimensionality reduction techniques to improve the performance of the previous mentioned *trace clustering* technique.

Despite the effort put in *trace clustering*, the motivation of the existent approaches relies on process discovery. It is known that large and complex event logs usually generate hard-to-read *spaghetti-like* models. With *trace clustering* the creation of several *sublogs* can be leveraged to discover process models out of cases with similar behaviour, thus reducing the process complexity and improving the model readability and comprehensibility. The approach proposed in this paper follows a different route, motivated by the developed *multiperspective conformance checking* algorithm. Instead of returning several *sublogs* that add up to the original event log, we propose the use of *clustering techniques* to generate a new, downsized event log that maintains the original process' knowledge.

8 Conclusions and future work

In this paper, a new data-aware conformance checking approach was proposed aiming at satisfying two main objectives: (a) avoid the generation of false-negative conformance errors in partially monitored processes; and (b) take advantage of all information related to the process model that was previously only available in natural language, and was then translated to a machine-understandable representation. The two-step conformance checking algorithm starts by converting a BPMN-E² model into a *Directly Follows Rules Model* (an extension of a *Directly Follows Model*) that is later used in the second phase to perform the conformance checking task effectively. During the conversion phase, the first objective is tackled by filtering the non-monitored activities. Additionally, the second objective is tackled in the second phase as the conformance checking algorithm produces a detailed report for each new element of the BPMN-E² model. Finally, a preliminary formula is proposed to compute the fitness of a model considering an event log and the set of conformance rules to be applied. To provide a more complete fitness value regarding both control-flow and data-flow, the combination of our approach with state-of-the-art alignment-based conformance checking algorithms is also encouraged.

An event log reduction algorithm was also proposed and developed following the trends in clustering techniques. These use state-of-the-art clustering algorithms to identify certain behaviour patterns on an event log and, mainly, to

discover groups of less cluttered and easier to read process models. In this work, however, clustering algorithms are used with the purpose of identifying homogeneous process behaviour between cases in order to downsize the event log by sampling the cases that better represent the process execution and removing those that prove to be redundant.

Both of the discussed approaches were developed using Python due to the vast amount of data mining related libraries and, especially, to the Process Mining library PM4Py. The solution was then used to analyse real event data related to a *Fine Management System*, highlighting the benefits of the approach. Synthetic event logs were also used to evaluate the performance of the conformance checking and the event log reduction algorithms.

There are still several ways to improve the currently developed features, such as: (a) improving the current visualisation module to enable an improved graphical representation, for example, through the development of a UI platform for enabling BPMN-E² creation and manipulation in an interactive fashion; (b) enabling the creation of custom conformance rules that do not fall under the built-in ones; (c) developing an enhanced API that facilitates the creation of independent DFRMs that can be used in conjunction with any other type of model notation; (d) finding a solution to overcome the limitation of DFM in handling parallel paths in the BPMN model; and (e) allowing online conformance checking by extending support for streaming event-data. Furthermore, the present work can be seen as the gateway to foster the use of BPMN-E² in the context of Process Mining, and, therefore, it is important to mention that the work focused only on one of the three types of Process Mining—conformance checking. Finally, there is still interesting and useful work to be done in respect to BPMN-E² notation and its usage in Process Mining techniques. Possible future work routes include the development of a *process discovery* technique to build BPMN-E² models starting from an attribute-rich event log; the development of a *process enhancement* technique to add BPMN-E² extension elements to an already existing BPMN model; or even, the adaptation of the present conformance checking technique to analyse streams of events in real-time.

Acknowledgements This work has been supported by FCT - Fundação para a Ciência e Tecnologia - within the R&D Units Project Scope: UIDB/00319/2020.

Author Contributions All authors have contributed to the paper in a fairly balanced way.

Funding Open access funding provided by FCTIFCCN (b-on). This work has been supported by FCT - Fundação para a Ciência e Tecnologia - within the R&D Units Project Scope: UIDB/00319/2020.

Availability of data and materials We have used a public event log - Road Traffic Fine Management Process - from an information system of the Italian police (available in <https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5>), and also synthetic event logs for testing purposes (not yet public).

Declarations

Conflict of interest There are no competing interests in this work.

Ethical approval Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. van der Aalst W (2016) Data Science in Action. In: Process mining. Springer: Berlin pp. 3–23
2. Grupe FH, Owrang MM (1995) Data base mining: discovering new knowledge and competitive advantage. *Inf Syst Manag* 12(4):26–31
3. Kubina M, Varmus M, Kubinova I (2015) Use of big data for competitive advantage of company. *Proc Econ Financ* 26:561–565
4. Kalenkova AA, van der Aalst WM, Lomazova IA, Rubin VA (2017) Process mining using BPMN: relating event logs and process models. *Softw Syst Model* 16(4):1019–1048
5. Van der Aalst W, Adriansyah A, Van Dongen B (2012) Replaying history on process models for conformance checking and performance analysis. *Wiley Interdiscip Rev Data Min Knowl Discov* 2(2):182–192
6. Van Der Aalst W, Adriansyah A, De Medeiros AKA et al (2012) Process mining manifesto. In: Lecture notes business information processing 99 LNBIP, pp. 169–194
7. Ramos-Merino M, Santos-Gago JM, Álvarez-Sabucedo LM, Alonso-Roris VM, Sanz-Valero J (2019) BPMN-E2: a BPMN extension for an enhanced workflow description. *Softw Syst Model* 18(4):2399–2419
8. Kalenkova AA, De Leoni M, Van Der Aalst WM (2014) Discovering, analyzing and enhancing BPMN models using ProM. In: CEUR workshop proceedings vol. 1295, pp. 36–40
9. Object Management Group (OMG) (2014) Business process model and notation , V2.0.2. Technical report January
10. White S (2004) Introduction to BPMN. *IBM Coop* 2:1–11
11. Munoz-Gama J (2014) Conformance checking and diagnosis in process mining: comparing observed and modeled processes. PhD thesis, Universitat Politècnica de Catalunya - BarcelonaTech
12. Adriansyah A, Van Dongen BF, Van Der Aalst WM (2011) Towards robust conformance checking. In: Lecture notes in business information processing. 66 LNBIP, pp. 122–133
13. van der Aalst W, van der Aalst W (2016) Data science in action. In: Process mining. Springer: Berlin pp. 3–23
14. Buijs J, Dongen van B, Aalst van der W (2013) On the role of fitness, precision, generalization and simplicity in process discovery. In: Meersman R (ed) On the move to meaningful internet systems: OTM 2012 (confederated international conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012. Proceedings, Part I). Lecture Notes in Computer Science, Germany, Springer pp. 305–322
15. Song M, Yang H, Siadat SH, Pechenizkiy M (2013) A comparative study of dimensionality reduction techniques to enhance trace clustering performances. *Expert Syst Appl* 40(9):3722–3737
16. Henry DB, Tolan PH, Gorman-Smith D (2005) Cluster analysis in family psychology research. *J Fam Psychol* 19(1):121–132
17. Antonenko PD, Toy S, Niederhauser DS (2012) Using cluster analysis for data mining in educational technology research. *Educ Tech Res Dev* 60(3):383–398
18. Omran MG, Engelbrecht AP, Salman A (2007) An overview of clustering methods. *Intell Data Anal* 11(6):583–605
19. Wilks DS (2011) Cluster analysis. *Int Geophys* 100:603–616
20. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. *ACM Comput Surv* 31(3):264–323

21. MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, pp. 281–296
22. Calheno R, Carvalho P, Lima SR, Henriques PR, Merino MR (2021) Multi-perspective conformance checking applied to bpmn-e². In: Rocha Á, Adeli H, Dzemyda G, Moreira F, Correia AMR (eds) Trends and applications in information systems and technologies - Vol. 2, WorldCIST 2021, Terceira Island, Azores, Portugal, 30 March - 2 April, 2021. Vol. 1366 of Advances in intelligent systems and computing., Springer pp. 394–404
23. Leemans SJ, Poppe E, Wynn MT (2019) Directly follows-based process mining: exploration & a case study. In: Proceedings—2019 international conference on process mining, ICPM, pp. 25–32
24. Biere A, Heule M, Van Maaren H, Walsch T, Barrett C, Sebastiani R, Seshia SA, Tinelli C (2008) Handbook of satisfiability satisfiability modulo theories. Technical report, OS Press
25. Sebastiani R, Armando A, Barrett C, Bozzano M, Bruttomesso R, Cimatti A, Franzen A, De Moura L, Ghilardi S, Griggio A, Krstic S, Nieuwenhuis R, Oliveras A, Ranise S, Roveri M, Strichman O, Tacchella A, Tinelli C (2007) Lazy satisfiability modulo theories important discussions with. *J Satisfiabil Boolean Model Comput* 3:141–224
26. de Moura L, Bjørner N (2008) Z3: an efficient SMT solver. In: Lecture notes in computer science. Springer-Verlag, pp. 337–340
27. Gan G, Ng MKP (2017) K-means clustering with outlier removal. *Pattern Recogn Lett* 90:8–14
28. Mannhardt F, de Leoni M, Reijers HA, van der Aalst WM (2016) Balanced multi-perspective checking of process conformance. *Computing* 98(4):407–437
29. Liu Y, Li Z, Xiong H, Gao X, Wu J (2010) Understanding of internal clustering validation measures. In: Proceedings—IEEE international conference on data mining, ICDM, pp. 911–916
30. De Leoni M, Van Der Aalst WM (2013) Data-aware process mining: discovering decisions in processes using alignments. In: Proceedings of the ACM symposium on applied computing pp. 1454–1461
31. De Leoni M, Van Der Aalst WM (2013) Aligning event logs and process models for multi-perspective conformance checking: an approach based on integer linear programming. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics) 8094 LNCS, pp. 113–129
32. Estañol M, Munoz-Gama J, Carmona J, Teniente E (2019) Conformance checking in UML artifact-centric business process models. *Softw Syst Model* 18(4):2531–2555
33. Valencia-Parra Á, Varela-Yaca AJ, Gómez-López MT, Carmona J, Bergenthum R (2021) Empowering conformance checking using big data through horizontal decomposition. *Inf Syst* 99:101731
34. De Leoni M, Van Der Aalst WM, Van Dongen BF (2012) Data- and resource-aware conformance checking of business processes. In: Lecture notes in business information processing 117 LNBIP, pp. 48–59
35. Ly LT, Rinderle-Ma S, Knuplesch D, Dadam P (2011) Monitoring business process compliance using compliance rule graphs. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics) 7044 LNCS (PART 1), pp. 82–99
36. Ly LT, Rinderle-Ma S, Dadam P (2010) Design and verification of instantiable compliance rule graphs in process-aware information systems. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics) 6051 LNCS, pp. 9–23
37. Musa A, Gunasekaran A, Yusuf Y (2014) Supply chain product visibility: methods, systems and impacts. *Expert Syst Appl* 41(1):176–194
38. Wang L, Ting JS, Ip W (2013) Design of supply-chain pedigree interactive dynamic explore (SPIDER) for food safety and implementation of hazard analysis and critical control points (HACCPs). *Comput Electron Agric* 90:14–23
39. Zhang K, Chai Y, Yang SX, Weng D (2011) Pre-warning analysis and application in traceability systems for food production supply chains. *Expert Syst Appl* 38(3):2500–2507
40. Herzberg N, Meyer A, Weske M (2013) An event processing platform for business process management. In: 17th IEEE international enterprise distributed object computing conference. IEEE pp. 107–116
41. Baumgrass A, Herzberg N, Meyer A, Weske M (2014) BPMN extension for business process monitoring. In: Enterprise modelling and information systems architectures-EMISA

42. Kirchner K, Herzberg N, Rogge-Solti A, Weske M (2012) Embedding conformance checking in a process intelligence system in hospital environments. In: Process support and knowledge representation in health care. Springer pp. 126–139
43. Friedenstab JP, Janiesch C, Matzner M, Muller O (2012) Extending BPMN for business activity monitoring. In: 2012 45th Hawaii international conference on system science (HICSS), IEEE pp. 4158–4167
44. Rodríguez A, Fernández-Medina E, Piattini M (2007) A BPMN extension for the modeling of security requirements in business processes. *IEICE Trans Inf Syst* 90(4):745–752
45. Object Management Group (2016) Decision model and notation (DMN) Version 1.1
46. Mannhardt F (2018) Multi-perspective process mining. PhD thesis, Mathematics and Computer Science Proefschrift
47. Saeedi K, Zhao L, Sampaio PRF (2010) Extending BPMN for supporting customer-facing service quality requirements. In: 2010 IEEE international conference on web services (ICWS), IEEE pp. 616–623
48. Braun R, Schlieter H, Burwitz M, Esswein W (2014) BPMN4CP: design and implementation of a BPMN extension for clinical pathways. In: 2014 IEEE international conference on bioinformatics and biomedicine (BIBM), IEEE pp. 9–16
49. Ramón-Stroppi LJ, Chiotti O, David-Villarreal P (2011) A BPMN 2.0 extension to define the resource perspective of business process models. In: XIV Iberoamerican conference on software engineering, pp. 25–38
50. Cao Y (2018) Attribute-driven hierarchical clustering of event data in process mining master thesis. PhD thesis, RWTH Aachen University
51. Song M, Günther CW, Van Der Aalst WM (2009) Trace clustering in process mining. In: Lecture notes in business information processing 17 LNBIP, pp. 109–120

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Rui Calheno¹ · Paulo Carvalho¹ · Solange Rito Lima¹ · Pedro Rangel Henriques¹ · Mateo Ramos Merino²

✉ Paulo Carvalho
pmc@di.uminho.pt

Rui Calheno
a78085@alunos.uminho.pt

Solange Rito Lima
solange@di.uminho.pt

Pedro Rangel Henriques
prh@di.uminho.pt

Mateo Ramos Merino
mateoramosmerino@gmail.com

¹ Centro Algoritmi, Universidade do Minho, Braga, Portugal

² Escola Enxeñaría Telecomunicación, Universidad de Vigo, Vigo, Spain