Assessing Process Mining Techniques: a Ground Truth Approach

Dominique Sommers, Natalia Sidorova, Boudewijn F. van Dongen
Eindhoven University of Technology, Mathematics and Computer Science, Eindhoven, the Netherlands
{d.sommers, n.sidorova, b.f.v.dongen}@tue.nl

2024 6th International Conference on Process Mining (ICPM) | 979-8-3503-6503-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICPM63005.2024.10680619

***Abstract*—The assessment of process mining techniques using real-life data is often compromised by the lack of ground truth knowledge, the presence of non-essential outliers in system behavior and recording errors in event logs. Using synthetically generated data could leverage ground truth for better evaluation. Existing log generation tools inject noise directly into the logs, which does not capture many typical behavioral deviations. Furthermore, the link between the model and the log, which is needed for later assessment, becomes lost.**

We propose a ground-truth approach for generating process data from either existing or synthetic initial process models, whether automatically generated or hand-made. This approach incorporates patterns of behavioral deviations and recording errors to produce a synthetic yet realistic deviating model and imperfect event log. These, together with the initial model, are required to assess process mining techniques based on ground truth knowledge. We demonstrate this approach with a conformance checking use case, focusing on (relaxed) systemic alignments to expose and explain deviations in modeled and recorded behavior. Our results show that this approach, unlike traditional methods, provides detailed insights into the strengths and weaknesses of process mining techniques, both quantitatively and qualitatively.

***Index Terms*—Evaluation, Validation, Synthetic process data, Realistic noise, Behavioral patterns**

I. Introduction

A process mining assessment method should include *validation* to check the result for correctness, quantitative and qualitative *evaluation* to assess the effectiveness and accuracy, *reliability* assessment to look into repeatability and reproducibility of the results, *robustness* assessment to assess the ability of the method to cope with variations in the data and noise, *performance* and *scalability* assessment to assess efficiency and ability to cope with large and complex logs and models, and *usability* assessment to evaluate interpretability of obtained results.

Already in 2008, Rozinat et al. identified the need for a common framework for evaluating process mining results [1]. They emphasized that such a framework should help researchers compare algorithm performance and end users to validate their results. They also stressed that such a framework should allow users to influence process and log characteristics and support the generation of "forbidden" scenarios as a complement to the actual execution log. Despite the progress in the process mining field and the availability of tools for model generation, simulation, and injection of noise into event logs, the challenges formulated in [1] still remain relevant.

We encountered these challenges when working on the NWO CERTIF-AI project, in which we developed alignment methods for the conformance checking of processes involving multiple entities, such as objects and resources performing different tasks, taking the interaction of these entities into account in the alignments. We obtained data from real-life processes with varying characteristics and noise levels. However, when evaluating our alignment methods to check whether the recorded behavior of a manufacturing process from our partner company Omron is compliant with a prescriptive model provided by process owners, we found it difficult to validate the results without major efforts from stakeholders and time-consuming manual inspection, making a large scale evaluation and validation practically impossible. The exposed individual deviations in process behavior can be explained in several ways by e.g., timing issues in the recording, multitasking by human operators, or operators temporarily switching roles.

To validate the results, we would need to have knowledge about the true causes of deviations, be it recording errors or behavioral outliers. To conduct a proper large-scale validation we would need multiple models with varying characteristics, multiple executions of those models with different violations of the prescribed behavior, and multiple event logs with recording errors. In addition to that we need an "omniscient" stakeholder, an "oracle" able to tell whether the deviation detected in the alignments is a true deviation and whether we provide a correct explanation of this deviation.

It is clearly infeasible to provide such a "ground truth oracle" when using real-life data, especially if the goal is to do experimentation on a large scale. Therefore, we resort to using synthetically generated process models and event logs with realistic characteristics. Several tools like PTAndLogGenerator [2], PUPRLE [3], and AIR-BAGEL [4] make steps towards this goal. However, these inject noise directly into simulated logs, which does not allow for generating data containing typical behavioral deviations observed in real-life processes and validating whether the results of the process mining methods can handle such deviations correctly.

In this paper, we propose an approach for generating synthetic data in which realistic noise is incorporated into a simulation model with the help of model transformations capturing patterns of both behavioral deviations and recording errors, resulting in an imperfect process model and an event log obtained for this model. Both the transitions of the model, representing "true" events and deviations, and the event

records in the log contain tags indicating the deviation type, thus allowing to implement an oracle aware of behavioral outliers and recording errors. We illustrate the use of this approach with an alignment validation example.

This paper is organized as follows. In Sec. II, we discuss related work. In Sec. III, we discuss the impact of the characteristics of real-life data on the assessment and pose our research question concerning how this can be mimicked synthetically, providing knowledge of the ground truth. We describe our method and discuss how its produced components contribute to our goal in Sec. IV. Sec. V demonstrates the approach showing validation results and generated insights. We discuss the implications of our work in Sec. VI.

II. Related Work

In this section, we give a brief overview of assessment practices in process mining with real-life event logs and synthetic data.

A. Assessment using real-life logs with unknown model

Evaluating process mining techniques using real-life data requires significant effort and active participation of data owners. The case study with a UWV event log in [5] is an excellent example of this type of work. Although such case studies provide invaluable insights and help identify strengths and limitations of process mining techniques, the proprietary nature of this data often prevents it from being shared with the process mining community for future research. The assessment of repeatability and reproducibility of the studies becomes out of reach.

An established practice for evaluating process mining techniques is to use open data sets provided in yearly Business Process Intelligence Challenges (BPIC) from 2011 to 2020, which originate from real-life processes and can be found in the 4TU Data Repository. Some of these data sets have become very popular, with BPIC 2012 being cited in nearly 500 papers and BPIC 2017 by almost 300, according to Google Scholar at the time of writing this paper. From the very first years, BPIC event data was realistically noisy in terms of both its recording and behavior [6], which made this data important for testing and comparing various techniques in real-life scenarios.

A complicating factor in assessing process discovery techniques with BPIC data is that the ground truth knowledge about the underlying process is missing, with only limited information available thanks to efforts in understanding the processes through (manual) analyses and consultations with domain experts. Assessing the quality of such process mining techniques as conformance checking or model repair using BPIC data remains a complex and not straightforward task, as the corresponding behavioral models are not known.

To address the question of whether process mining methods work on real-life small, incomplete event logs, [7] developed an evaluation framework that reduces event logs and generates small event logs by removing traces either randomly or along the time dimension, producing training and test logs. Removal of traces can lead to side effects related to the workload of resources, case interactions, etc., making the log less realistic. In a sense, this approach moves towards evaluation with synthetic data.

B. Assessment with synthetic data

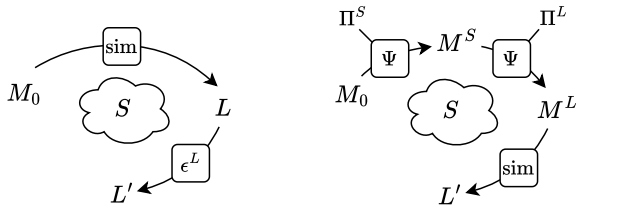
Besides real-life data, synthetic data is used to provide datasets for evaluating process mining methods. A number of tools were developed for this purpose. *PTAndLogGenerator* (PTALG) described in [2] is a tool that allows to generates a population of non-structured process models with user-defined probabilities for sequence, choice, parallel, and loop structures. Corresponding event logs consisting of single object traces are obtained by simulation. 'Noise' is imputed after simulation directly into traces and includes missing head, missing body (episode), missing tail, order perturbation, and the introduction of additional activities, adopting the definition of noise from [8], additionally incorporating a randomly generated decision model and data attributes.

In [3], a PURPose-Guided Log gEneration (PUPRLE) framework is proposed to produce event logs with different properties for targeting different mining purposes. It takes a process model, not restricted to its modeling language, to simulate an event log through (guided) execution of the model. Depending on the purpose of the synthetic data, the simulated event log can be attributed with different characteristics. E.g., for its use in process discovery and conformance checking, the resulting recorded behavior includes some predefined infrequent behavior, and noise, as defined by [8], respectively. Note that this noise is again limited to the same simple log manipulations as in PTALG, and is only imputed after simulation. Furthermore, the input process model represents the actual behavior of the process, i.e., it includes behavioral outliers contributing to infrequent behavior.

AIR-BAGEL [4] is an interactive tool designed to inject pseudo-real anomalies into event logs by associating them with specific root causes, such as resource behavior or system malfunctions, using a probabilistic mechanism for resource and systems errors, resulting in e.g. skipping a step or rework. The tool generates logs augmented with labels and attributes indicating the corresponding anomaly types, enabling the evaluation of event log cleaning methods.

Figure 1a provides a simplified illustration for these approaches where a process model M_0 is assumed to describe the true behavior of the process, and used to simulate a "clean" event log L. Noise is imputed only afterwards by a log manipulation function ϵ^L.

The availability of such tools facilitates large-scale evaluations of process mining techniques. For example, Van Houdt et al. [9] generated 400 artificial event logs using PTALG to generate models, BPSimPy to populate models with simulation parameters like the total simulation length, and L-Sim simulator to generate low-level event logs. The empirical evaluation of unsupervised log abstraction techniques presented in [9] primarily focuses on the precision and fitness of discovered models and provides interesting insights related to the influence of abstraction techniques used on the balance between



(a) Synthetic process data generation methods with noise injection function ϵ^L

(b) Generation of synthetic process data with behavioral and recording deviation patterns Π^S and Π^L and model transformation function Ψ

Fig. 1: Approaches to generating synthetic data (model M and log L) for an underlying process S

fitness and precision. They note that increasing the scale of experimentation allowed them to obtain more nuanced results.

III. Requirements to synthetic process data

To conduct an assessment of process mining techniques on synthetic but realistic data, we should be able to generate synthetic process data including (1) process models with different process characteristics, like the degree of parallelism or non-determinism, as they may unknowingly influence the results, (2) variation in deviations/noise, both in process execution and its recording, consistent to real-life deviation patterns, and (3) a ground truth which provides a target function for the assessment problem at hand.

Process models representing processes with different characteristics can be generated using existing model generation tools, like PTALG. In this paper, we address requirements (2) and (3) only, assuming that we already have models representing the system, automatically or manually generated.

A. Characteristics of real-life processes and their data

We need mechanisms to generate data with typical *recording errors* and *behavioral outliers*, with the type of deviation clearly indicated for later use in the assessment.

We consider recording errors in event logs as records misrepresenting the actually executed behavior of the process. In [6], *recording errors* are subdivided into four categories of issues, where data could be (1) missing, (2) incorrect, (3) imprecise, and (4) irrelevant. Each category could apply to various elements of the event log, e.g., events, event attributes, relations between attributes and events, and between events themselves. Possible sources of recording errors are faulty logging mechanisms, either automatic or manual, with unsynchronized clocks, too coarse timestamps, data corruption, and filtering and aggregation methods.

A process model ties together a set of modeling patterns [10] to define the "main" behavior of the process. The interpretation of the "main behavior" depends on the process mining task. It could be the expected behavior, the "happy flow" behavior, or frequent behavior. *Behavioral outliers* refer to behavior that deviates from the *main behavior* of the process. There could be several reasons for violating the modeled behavior, e.g., people not following guidelines, or encountering rarely seen cases not taken into account in the model, thus choosing different behavioral patterns than the ones imposed by the model.

Process mining techniques should be robust to recording errors and should be able to handle behavioral deviation appropriately (defined e.g., based on their frequencies), depending on the use case. Take for example process discovery. When the discovered process model is to be used as a handbook for employees, the goal might be to capture only the frequent behavior, whereas when the process model is to be used as a digital twin, the goal is to capture all possible behavior.

When resorting to the use of synthetic data, these deviation characteristics should be mimicked as closely as possible to represent real-life data. Model-log pairs that can be obtained by the approach illustrated in Fig. 1a are not sufficient, as (1) the model of the "real system" is not provided: the base model does not show behavioral deviations, or they are included in the base model and not distinguishable from regular behavior, (2) the log manipulations are performed irrespectively of the behavior of the process, therefore losing the link between model and log.

B. The need for ground truth

Dealing with these categories of errors and deviations is a challenge for process mining methods in real-life settings in general. This holds even more so for their assessment, as the explanations for causes of deviations may be ambiguous or missing altogether. For the evaluation of process discovery techniques, conformance metrics allow for measuring the quality of a process model where the ground truth model is unknown. However, this assumes the event log is free of recording errors, since differentiating them from behavioral outliers may be impossible due to ambiguity in their causes. While recording errors should ideally be filtered out completely, (some) behavioral outliers should end up in the behavior of the discovered process model. Therefore, for validation, being able to make this distinction is required. Similarly, for conformance checking, assessing whether deviations are correctly exposed and explained, having knowledge of the causes is essential for constructing the target.

Without going into detail, we argue that other PM problems, e.g., decision mining, model or log repair, bottleneck detection, and predictive and prescriptive process monitoring, suffer from the same complications. Therefore researchers are forced to resort to assessing PM techniques using synthetically generated data.

This brings us to our research question (RQ):

RQ: How can we set up an experiment with realistically noisy process data where the ground truth is known to allow for a complete assessment?

IV. Generating Process Data with Ground Truth

Depending on the goal of a PM algorithm/tool, and its input and output, its assessment may require a process model, an

event log, and the ground truth containing information about both behavioral and recording deviations that are to be used in assessing the quality of the produced results.

In this section, we propose an evaluation framework for generating such process data using a collection of deviation patterns and corresponding model transformations to build the deviation into an initial process model. We first give a high-level overview of the framework, after which we discuss deviation patterns and model transformations.

A. Framework design

Our framework is schematically depicted in Figure 1b. M_0 denotes a base model representing the expected behavior of the process, M^S serves as the true model of the process, capturing behavioral deviations, and M^L extends this model by imitating faulty logging mechanisms in the model. A simulation of M_0 results in an "exemplary" log showing the expected behavior of a process. A simulation of M^S gives a "clean" event log, showing the "real" behavior of a process. Simulating M^L results in log L' capturing the real behavior with realistic recording errors. This log L' is the target of our framework.

M_0, M^S, M^L, L' together with the links between them provide the ground truth about the process and its execution, including behavioral characteristics of the process and recording characteristics of the event log as well as its completeness with respect to the process (M^S) and the model (M_0).

The base model M_0 might describe an existing process, be a hypothetical process designed manually with certain characteristics in mind, or it can be generated automatically by a model generation tool. Recall that M_0 merely serves as a base process that does not represent the "real" process execution. It can serve as e.g., a normative model in the assessment of conformance-checking techniques, or a "typical flow" model in the assessment of process discovery techniques.

We start with a base process model M_0 and a set Π of deviation patterns. Deviations, either recording or behavioral, are modeled as templates. A model transformation function Ψ defines how the behavior modeled in deviation pattern $\pi \in \Pi$ can be added to a model. The deviating behavior is incorporated into the base process model, resulting first in model M_S of process executions with behavioral deviations, and subsequently in model M_L incorporating log recording deviations on top of that. By simulating the resulting model M_L, we acquire a realistic event log L' and the knowledge of all deviations that took place, since the simulated events can be linked back to transition firings from M^L, which in their turn can be linked to the transitions of M_S and M_0, or are labeled as corresponding deviations.

The set Π of deviations consists of subsets Π^S with behavioral deviation patterns and Π^L with recording error patterns. A model transformation function Ψ takes a model M, a deviation pattern $\pi \in \Pi$, and a function h mapping elements of π to the elements of M in order to indicate where the deviation should occur in the model. The model transformation series starts with M_0, and after applying the first transformation, applies the next transformation to the obtained $\Psi(M_0, \pi_1, h_1)$, etc., finally leading to the model M^S of the actual process execution. Similarly, we start with M^S to apply patterns from Π^L and obtain M^L after a series of model transformations. When simulated, M^L produces the event log L', encoding the complete information about the transitions that generated events.

Only partial information on the events (e.g. transition labels) is given to a process mining technique to be assessed. The rest of the information constitutes the ground truth knowledge to be used to compute quality metrics in the assessment and obtain qualitative insights.

We generalized and extended our first implementation of this framework that was presented in [11]. This new implementation supports an extendable set of deviation patterns and it can be found in gitlab.com/dominiquesommers/mira/-/tree/main/mira/simulation. The implementation is based on the use of typed Petri nets with Identifiers (t-PNIDs) [12], allowing to generate rich logs of processes with event records including information about resources that participated in their execution. This framework naturally extends to generate logs of stochastic, temporal, and object-centric processes.

B. Usage in assessments

To use the evaluation framework, assessors must define a ground truth function gt for their PM problem and assessment tasks. This function transforms the combination of M_0, M^S, M^L, and L' to the target result for that process mining technique. Further on, the assessors are to choose a distance function d to compare the result produced by the process mining technique to the ground truth target. Generically,

$$d\left(f(M_0, L'), \mathrm{gt}^f(M_0, L', M^S, M^L)\right)$$

measures the quality of a PM method f, using distance function d and ground truth function gt^f.

Evaluation and validation of *process discovery* (PD) and *process repair* techniques can be performed by comparing discovered models to either M_0, M^S, and/or M^L, depending on the application at hand. E.g., the goal can be to discover an interpolation of M_0 and M^S, including frequent behavioral deviations but ignoring the infrequent ones and all the recording errors. The target process model is defined by M_0 and a subset $\Pi^S_x \subseteq \Pi^S$ of behavioral deviations. The similarity between a discovered model $PD(L')$ and $\mathrm{gt}^{PD}(M_0, M^S, M^L, L')$ can be computed e.g., using model similarity [13].

The goal of *conformance checking* (CC) with $f = CC$ is to identify and explain deviations in the recorded behavior from the behavior captured by a normative model. A CC technique takes M_0 and L' as input e.g. to compute an alignment of the actual behavior of the process and the modeled behavior. The ground truth target can be defined as the optimal alignment, interpreting events corresponding to the firing of original transitions of M_0 as synchronous moves and events corresponding to the transitions being part of the deviation patterns as log moves or model moves accordingly.

The similarity between the alignment $CC(M_0, L')$ produced by a conformance checking technique and the ground

TABLE I: Deviation patterns.

Recording error patterns		Behavioral deviation patterns	
Missing event	R01	B01	Switching correlation
Incorrect event	R02	B02	Multitasking
Missing object	R03	B03	Skipping activity
Incorrect object	R04	B04	Neglecting object(s)
Missing position	R05	B05	Overtaking in queue
Incorrect position	R06	B06	In/decreasing capacity
		B07	Switching roles
		B08	Same resource on 4-eyes principle
		B09	Different resource on resource memory
		B10	Ignoring batching

truth alignment $\mathrm{gt}^{CC}(M_0, M^S, M^L, L')$ can be computed e.g., by sequence or graph edit distance, depending on the formalization used to model alignments [14], [15]. Note that this is an abstract definition of gt^{CC}. We concretize this in Sec. V, where we use our proposed evaluation framework to evaluate the explainability of multi-object alignments [16] and relaxed multi-object alignments [17].

C. Deviation patterns

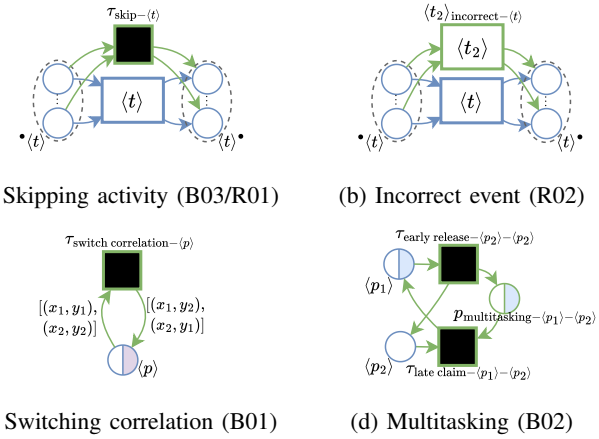
In Tab. I, we list a selection of deviation patterns that are currently implemented in our framework. $B01 - B10$ are behavioral deviation patterns originating from the patterns discussed in [10]. $R01 - R06$ forms a subset of recording errors presented in [6].

Take e.g., the recording error pattern $R01$, describing a missing event, where an activity is executed but failed to be recorded. Skipping an activity during the execution of a process is a behavioral deviation being a counterpart of a missing event: In both cases, the corresponding event will be missing in log L', although the causes would differ. The links between recording errors and their counterparts in the list of behavioral deviations are shown by the connecting lines.

A deviation pattern consists of a behavioral description and an abstract process model fragment. The abstract process model fragment serves as a blueprint for incorporating the deviating behavior into a process model. Formally, a *deviation pattern* is an abstract process model $\bar{M}$ describing deviating behavior in relation to some process behavior. Abstract means here that certain model elements are represented by wildcards acting as placeholders for specific elements of the model to which the deviation is to be added.

To illustrate and clarify the use of patterns, we elaborate on the modeled deviation patterns for a few examples from Tab. I, using t-PNIDs as the modeling formalism. This formalism allows for modeling intricate behavior of several interacting objects, corresponding to patterns described in Sec. III.

Deviation pattern B03, *skipping activity*, can be modeled by the abstract t-PNID shown in Fig. 2a, with a matched transition $\langle t \rangle$ from the base model, and its incoming and outgoing places with corresponding arcs. Additionally, a newly created silent transition $\tau_{\text{skip-}\langle t \rangle}$ inherits the same incoming and outgoing places creating the corresponding arcs. The place types and the



(a) Skipping activity (B03/R01) (b) Incorrect event (R02)

(c) Switching correlation (B01) (d) Multitasking (B02)

Fig. 2: t-PNID modeling blueprints for four deviation patterns. Blue and green elements respectively correspond to matched and newly created elements.

arc labels are inherited from the base model, and the initial and final markings are both empty. R01, *missing event*, is defined similarly, with $\tau_{\text{missing-}\langle t \rangle}$ instead of $\tau_{\text{skip-}\langle t \rangle}$, to differentiate between the two in the log obtained by simulation.

The recording error of *incorrect event* (R02) can be modeled by a t-PNID as shown in Fig. 2b, where an additional transition $\langle t_2 \rangle_{\text{incorrect-}\langle t_1 \rangle}$ is a copy of existing transition $\langle t_1 \rangle$, labelled with $\langle t_2 \rangle$.

Fig. 2c shows the behavioral deviation of *switching the correlation* between two multicolored tokens in a place (B01), through a silent transition and labeled arcs.

Lastly, the behavioral deviation of *multitasking* (B02) is illustrated in Fig. 2d, where the correlation between an object and a resource can be temporarily destroyed to make the resource available for other tasks ($\tau_{\text{early release-}\langle p_1 \rangle\text{-}\langle p_2 \rangle}$). $\tau_{\text{late claim-}\langle p_1 \rangle\text{-}\langle p_2 \rangle}$ can subsequently repair the correlation, to continue work on the first object.

Note that we assume that deviations, both for behavioral outliers and recording errors, can be modeled abstractly in the same formalism as the base model.

Our framework implements all patterns described in Tab. I, and it is not restricted to those. The list can be extended with additional pattern descriptions and corresponding definitions. All provided patterns are modeled in a way that the model transformation is additive in behavior, i.e., behavior described in the pattern is added to the base model without eliminating any behavior that was captured in the model. For this reason, multiple deviation patterns do not interfere with each other. The created elements have distinct names. This is a necessary property to ensure that the resulting process model allows sufficient freedom to define ground truth functions for different process mining tasks.

D. Model transformations

Behavior from a deviation pattern π, described by an abstract process model M_π, can be incorporated into a process

model M, through a model transformation function Ψ which uses M, M_π, and an injective mapping function h mapping the wildcards from π to elements in M.

Our definition for Ψ is derived from Petri net transformations as discussed in [18] and is defined as follows:

$$\Psi(M, \pi, h) = M \cup h(\pi) \tag{1}$$

where $h(\pi)$ is a morphism from the abstract process model fragment π to a concrete process model fragment in the space of M, with *matched elements* referring to elements in M. Through Ψ, $\Psi(M, \pi, h)$ additionally contains the *created components* of π, providing the added behavior.

E. Simulation

The event log L', which is the last element generated by the framework, describes the recorded behavior of the process. L' is obtained by simulation of the process model M^L, which represents the real behavior of a process including behavioral outliers (by Π^S) and faulty recording mechanisms (by Π^L), deviating from the expected behavior from M_0.

The simulation module is an interchangeable part of the framework which in our provided implementation is a play out of M^L, allowing for repeated firings of transitions, either through being enabled, or scheduled in the form of arrivals or predetermined schedules.

Probabilities of transition firings are modeled by adding weights to transitions, by sampling from a weighted distribution for all enabled transitions with corresponding bindings.

The simulated behavior of the process model M^L resembles real-life behavior by design, as it incorporates both behavioral deviations and recording errors.

V. Evaluating the interpretability of (Relaxed) multi-object alignments

As mentioned in Sec. II, alignments are used in conformance checking to expose deviations and provide possible explanations for discrepancies between a process model and a recorded event log. Here, we demonstrate our ground truth assessment approach for qualitative validation and evaluation of three alignment techniques on a synthetically designed process.

A. Experimental setup

The assessment question (AQ) we aim to answer is twofold:

AQ1: Are behavioral outliers and recording errors detected correctly?
AQ2: How well do the generated alignments reflect the ground truth explanations?

We start by designing a base process model M_0 describing the expected behavior of a package delivery process. An abstract, simplified overview of the process is illustrated in Fig. 3, starting with the choice of *home* or *depot* delivery, after which the package queues for a *warehouse employee* (w) to *pick* and *load* it into a *truck* (v). In case of home delivery, a *deliverer* (d) drives off and *rings* a door after which he continues to either immediately hand over the package (deliver home), or deliver it at the corresponding *depot* after *registration*, where it is left for *collection*. Alternatively, for depot delivery, 'ringing' and therefore also 'deliver home' is omitted in the subprocess. This process is making use of several modeling patterns, like choices, parallelism, queueing, batching, capacity, resource memory, and (continuous) correlation.

Next, we select a set of deviation patterns and apply them to M_0, providing M_S and M_L through Ψ. This set contains abstract deviation patterns from Tab. I applied on the concrete modeling patterns in M_0, as shown in Fig. 3 by the purple and yellow thunderbolts, respectively indicating *behavioral outliers* and *recording errors*.

The following behavioral deviation patterns are applied: B05 (overtaking) to the picking queue, B07 (switching role) to the deliverer taking on the role of a warehouse employee, B03 (skip activity) to ringing, modeling behavior where e.g., the door was already opened upon arrival, B02 (multitasking) on the deliverer, modeling interruption of a delivery, and B09 (different resource memory) where the package is delivered to a different depot as where it is registered.

For recording errors, the following patterns are applied: B02.1 and B02.2 (incorrect event) recording depot delivery when home delivery was intended, and vice versa, R01 (missing event) and R03 (missing object) for loading in a truck, e.g., due to temporary connection failure of a recording device, R04.1 (incorrect object) of the deliverer when ringing, e.g., due to not logging out by the deliverer on the previous shift, and R05 (missing position) of deliver depot and collect, e.g., due to coarse timestamp logging.

By simulation of M_L, we generated event log L'. To isolate deviation patterns, we separated the event log into sublogs each containing two package orders delivered using the same delivery van object. The ground truth oracle provides explanations by knowledge of which transitions have fired in the simulation to produce the events in L'.

M_0 and the sublogs in L' serve as input for the three alignment methods we consider in this assessment. These are (1) traditional *per-object alignments* (γ^o), operating on each object in isolation, both in event log and model, ignoring any interaction between objects [19], (2) *systemic* alignments (γ), aligning the event log in its entirety to the process model taking into account violations [16], and (3) *relaxed systemic* alignments ($\tilde{\gamma}$), extending their regular counterpart by allowing for relaxations of objects' interactions both in the log and model [17].

Each method takes an event log being a partially ordered set and a normative model as input. The log used for the alignments is a projection of L' with some types of deviating events (like skipped steps or missing events) removed from it. The produced alignment contains synchronous moves (on which the logg and model agree), log moves (indicating that an activity from the log cannot be mimicked in the model) and model moves (indicating which events required by the model

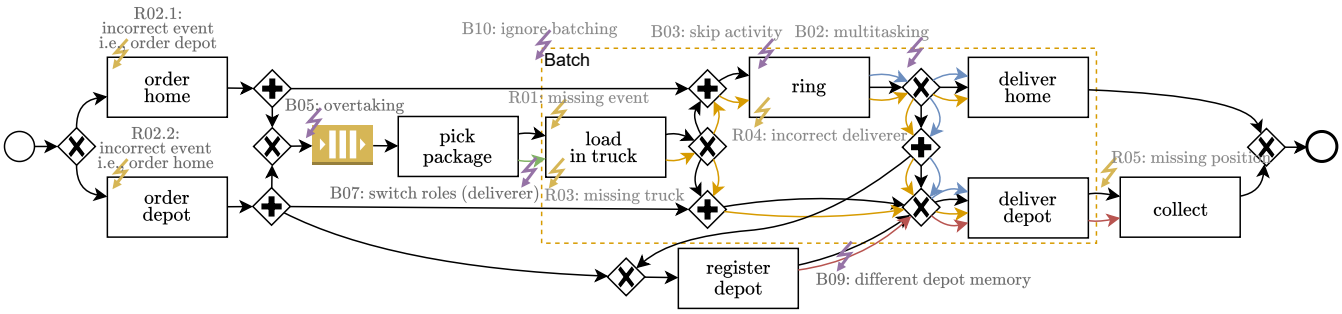


Fig. 3: Abstract process model for the package delivery process, including modeled behavior of *packages* (black), *FIFO queue* (yellow), *warehouse employees* (green), *delivery vans* (orange), *deliverers* (blue), and *depots* (red). Additionally, the added behavioral outliers (purple thunderbolts) and recording errors (yellow thunderbolts) are annotated.

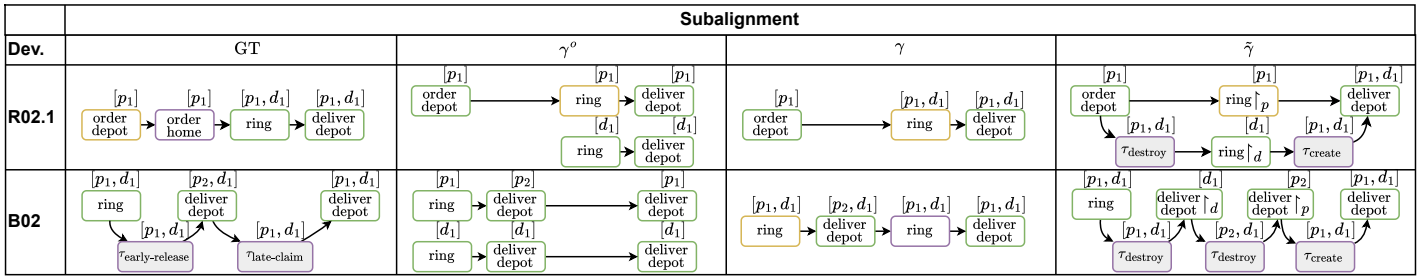


Fig. 4: Ground truth (GT) and computed (γ^o, γ, and $\tilde{\gamma}$) sub-alignment results for deviation patterns R02.1 and B02. Log, model, and synchronous moves are colored yellow, purple, and green respectively. For clarity, truck and depot objects are omitted.

TABLE II: Experiment results showing the deviating objects (and those affected by the deviation) for each pattern, for the ground truth (GT) and detected by alignment methods: per-object (γ^o), systemic (γ), and relaxed systemic ($\tilde{\gamma}$).

	Deviating object(s)					Deviating object(s)			
Dev.	GT	γ^o	γ	$\tilde{\gamma}$	**Dev.**	GT	γ^o	γ	$\tilde{\gamma}$
R02.1	$\{p\}$	$\{p\}$	$\{p,d\}$	$\{p\}$	**B05**	$\{q\}_{(\{p\})}$	$\emptyset$	$\{p,q,w\}$	$\emptyset_{(\{p,q\})}$
R02.2	$\{p\}$	$\{p\}$	$\{p,q\}$	$\{p\}$	**B07**	$\{d\}_{(\{p,v\})}$	$\{d\}$	$\{p,d,v\}$	$\{d\}_{(\{p,v\})}$
R01	$\{p,v,e\}$	$\{p,v,e\}$	$\{p,v,e\}$	$\{p,v,e\}$	**B10**	$\{v\}_{(\{p,d\})}$	$\emptyset$	$\{p,d,v,w\}$	$\emptyset_{(\{v,p,d\})}$
R03	$\{v\}_{(\{p,e\})}$	$\{v\}$	$\{p,v,e\}$	$\{v\}_{(\{p,e\})}$	**B03**	$\{p,d\}$	$\{p,d\}$	$\{p,d\}$	$\{p,d\}$
R04	$\{d\}_{(\{p\})}$	$\{d\}$	$\{p,d\}$	$\{d\}_{(\{p\})}$	**B09**	$\{w\}_{(\{p,d\})}$	$\{w\}$	$\{w,p\}$	$\{w\}_{(\{p,d\})}$
R05	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	**B02**	$\{d\}_{(\{p,v\})}$	$\emptyset$	$\{p,d\}$	$\emptyset_{(\{p,v,w\})}$

are missing in the log) can be compared to the log L'.

B. Results and discussion

The detected deviations for each sublog are shown in Tab. II. Each row in the two tables corresponds to a sublog. The Dev. column indicates the deviation used in the corresponding sublog. The place of the deviations in the model is shown in Fig. 3. The cells of the table specify the set of objects included in log and model moves of the corresponding alignments, and in addition the set of affected objects in brackets where applicable.

For recording errors (left), each deviation was detected by each method, however, there is a difference in the involved objects when the deviation occurs on object level rather than event level. For the single-object alignment γ^o, we see only the deviation's responsible object, for the system alignment γ, we see all objects involved in the corresponding activity, and for the relaxed systemic alignment $\tilde{\gamma}$, we see a combination of the two, correctly separating responsible objects from affected objects. The last recording error (R05) is not considered a deviation in the context of alignments. From the computed alignments for R02.1 (c.f., Fig. 4), we see that none of the methods resolve the deviation correctly, as all conclude that the *ring* event is incorrectly logged instead of the *order depot* event. The subalignments clearly show how each method distinguishes between the involved objects.

For behavioral outliers (right), the two deviations B07 and B03 occur on the level of isolated activity or object and are detected similarly as the recording errors discussed above. The others are not detected by single-object alignments, as they correspond to deviations in objects' interactions, which are not considered. Except for B09, where the executed behavior of both depot objects is incomplete according to the model.

Systemic alignments γ expose all deviations, again noting all involved objects without indication of the object(s) responsible for the deviation. This distinction is made by relaxed alignments $\tilde{\gamma}$ for deviations B07 and B03, while for the remaining deviations, the involved objects are detected without log and (labeled) model moves.

From the alignments computed for B02 (c.f., Fig. 4), we can see the consequence of ignoring correlations of objects, as the single-object alignments γ^o show no deviations. This is

resolved in the systemic alignment γ. However, its explanation differs from the deviation pattern used in L', contrary to the relaxed systemic alignment $\tilde{\gamma}$, which includes (relaxed) synchronous moves for each recorded event and alterings of correlations. Note that these alignments still allow for variance in their interpretation, e.g., the separated synchronous moves for deliver depot and the non-matching silent model moves.

In general, to answer AQ1 and AQ2, we see that single-object alignments can pinpoint deviations on the level of the responsible object, but only if they occur within the workflow of the particular object. Systemic alignments are robust in detecting the deviations but fail to provide details for the responsible objects. Relaxed systemic alignments do both, however, not always provide the correct explanation. This helps to outline the directions for future work in this area. Having the ground truth knowledge provided by our framework was indispensable for conducting this assessment.

VI. Conclusion and Future Work

Where real-life process data provides for valuable assessments of process mining techniques used in realistic scenarios, the absence of an omniscient stakeholder and specific process characteristics may limit the validity and generalizability of the assessment results. The use of synthetic data allows extending the scale of experimentation and creating an oracle generating the ground truth to be used in assessment.

The approach proposed in this paper makes use of deviations in terms of behavioral outliers and recording errors known from the literature and captures them as modeling patterns and corresponding model transformations. Starting from a base model, we generate separate models for an executed process and for recorded process behavior, as well as a synthetically generated yet realistically imperfect event log. This approach is limited by the assumption that the deviations can be modeled using the same formalism as the initial model, and by the dependency on the simulation method and parameters regarding the frequencies of the incorporated deviations, potentially affecting the realism of the dataset. The list of deviations described (and implemented) in this work is inherently incomplete and is designed to be extended depending on the domain. E.g., it naturally extends to the stochastic and temporal aspects of the process, which can either be modeled in patterns or the simulation module directly.

Through a demonstration, we show that this approach provides validation and qualitative insights into the strengths and weaknesses of different alignment techniques for the detection and explanation of deviations in modeled and recorded behavior. The results can be differentiated per deviation type, i.e., recording errors and behavioral outliers, as well as per deviation itself, and tailored towards domain-specific parts of the process.

Whereas in this work, we focused on the *qualitative* aspect of the *evaluation*, this approach trivially extends to the *quantitative* aspect, by providing a broader dataset with varying frequencies of deviating behavior and analyzing the corresponding statistics of the results. Subsequently, it allows for a *reliability*, *robustness*, *performance*, and *usability* assessment, quantitatively as well as qualitatively, providing results with regard to characteristics of the process, i.e., in terms of domain-specific (expected) deviating behavior.

Acknowledgements. This work is done within the project "Certification of production process quality through Artificial Intelligence (CERTIF-AI)", funded by NWO (project number: 17998).

References

[1] A. Rozinat, A. K. A. de Medeiros, C. W. Günther, A. J. M. M. Weijters, W. M. P. van der Aalst, The need for a process mining evaluation framework in research and practice, in: A. ter Hofstede, B. Benatallah, H.-Y. Paik (Eds.), Business Process Management Workshops, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 84–89.

[2] T. Jouck, B. Depaire, Generating artificial data for empirical analysis of control-flow discovery algorithms: A process tree and log generator, Business & Information Systems Engineering 61 (2019) 695–712.

[3] A. Burattin, B. Re, L. Rossi, F. Tiezzi, A purpose-guided log generation framework, in: International Conference on Business Process Management, Springer, 2022, pp. 181–198.

[4] J. Ko, J. Lee, M. Comuzzi, AIR-BAGEL: An interactive root cause-based anomaly generator for event logs., in: ICPM Doctoral Consortium/Tools, Vol. 2703, CEUR Workshop Proceedings, 2020, pp. 35–38.

[5] M. Dees, M. de Leoni, F. Mannhardt, Enhancing process models to improve business performance: A methodology and case studies, in: H. Panetto, C. Debruyne, W. Gaaloul, M. Papazoglou, A. Paschke, C. A. Ardagna, R. Meersman (Eds.), On the Move to Meaningful Internet Systems. OTM 2017 Conferences, Springer International Publishing, Cham, 2017, pp. 232–251.

[6] R. J. C. Bose, R. S. Mans, W. M. v. Aalst, Wanna improve process mining results?, in: 2013 IEEE symposium on computational intelligence and data mining (CIDM), IEEE, 2013, pp. 127–134.

[7] M. Käppel, S. Jablonski, S. Schönig, Evaluating predictive business process monitoring approaches on small event logs, in: A. C. R. Paiva, A. R. Cavalli, P. Ventura Martins, R. Pérez-Castillo (Eds.), Quality of Information and Communications Technology, Springer International Publishing, Cham, 2021, pp. 167–182.

[8] C. W. Günther, Process mining in flexible environments (2009).

[9] G. Van Houdt, M. de Leoni, N. Martin, B. Depaire, An empirical evaluation of unsupervised event log abstraction techniques in process mining, Information Systems 121 (2024) 102320.

[10] N. Russell, A. H. Ter Hofstede, W. M. Van Der Aalst, N. Mulyar, Workflow control-flow patterns: A revised view (2006).

[11] D. Sommers, D. V. Varadarajan, N. Sidorova, Trident: Generating noisy synthetic processes with ground truth (2023).

[12] J. M. E. van der Werf, A. Rivkin, A. Polyvyanyy, M. Montali, Data and process resonance: Identifier soundness for models of information systems, in: International Conference on Applications and Theory of Petri Nets and Concurrency, Springer, 2022, pp. 369–392.

[13] R. Dijkman, M. Dumas, B. Van Dongen, R. Käärik, J. Mendling, Similarity of business process models: Metrics and evaluation, Information Systems 36 (2) (2011) 498–516.

[14] L. Yujian, L. Bo, A normalized Levenshtein distance metric, IEEE transactions on pattern analysis and machine intelligence 29 (6) (2007) 1091–1095.

[15] X. Gao, B. Xiao, D. Tao, X. Li, A survey of graph edit distance, Pattern Analysis and applications 13 (2010) 113–129.

[16] D. Sommers, N. Sidorova, B. F. v. Dongen, Aligning event logs to Resource-Constrained Petri nets, in: International Conference on Applications and Theory of Petri Nets and Concurrency, Vol. 13288, Springer, 2022, pp. 325–345.

[17] D. Sommers, N. Sidorova, B. F. v. Dongen, Conformance checking with model projections - rethinking log-model alignments for processes with interacting objects, in: International Conference on Applications and Theory of Petri Nets and Concurrency, Vol. 14628, Springer, 2024.

[18] H. Ehrig, J. Padberg, Graph grammars and petri net transformations, in: Advanced Course on Petri Nets, Springer, 2003, pp. 496–536.

[19] J. Carmona, B. van Dongen, A. Solti, M. Weidlich, Conformance checking, Switzerland: Springer 56 (2018) 12.