EasyChair Preprint

№ 5841

A Method for Debugging Process Discovery Pipelines to Analyze the Consistency of Model Properties

Christopher Klinkmüller, Alexander Seeliger, Richard Müller, Luise Pufahl and Ingo Weber

June 17, 2021

A Method for Debugging Process Discovery Pipelines to Analyse the Consistency of Model Properties

Christopher Klinkmüller[1], Alexander Seeliger[2], Richard Müller[3], Luise Pufahl[4], and Ingo Weber[4]

[1] CSIRO Data61, Sydney, Australia, christopher.klinkmueller@data61.csiro.au
[2] TU Darmstadt, Germany, seeliger@tk.tu-darmstadt.de
[3] Leipzig University, Germany, rmueller@wifa.uni-leipzig.de
[4] Chair of Software and Business Engineering,
Technische Universitaet Berlin, Germany, ⟨firstname⟩.⟨lastname⟩@tu-berlin.de

Abstract. Event logs have become a valuable information source for business process management, e.g., when analysts discover process models to inspect the process behavior and to infer actionable insights. To this end, analysts configure discovery pipelines in which logs are filtered, enriched, abstracted, and process models are derived. While pipeline operations are necessary to manage log imperfections and complexity, they might, however, influence the nature of the discovered process model and its properties. Ultimately, not considering this possibility can negatively affect downstream decision making. We hence propose a framework for assessing the consistency of model properties with respect to the pipeline operations and their parameters, and, if inconsistencies are present, for revealing which parameters contribute to them. Following recent literature on software engineering for machine learning, we refer to it as *debugging*. From evaluating our framework in a real-world analysis scenario based on complex event logs and third-party pipeline configurations, we see strong evidence towards it being a valuable addition to the process mining toolbox.

Keywords: Process Mining, Discovery, Uncertainty & Sensitivity Analysis

1 Introduction

Historic process information from *event logs* enables analysts to derive business process insights using *process mining* [1]: *process discovery* [5,19] infers process models from the recorded behavior, *conformance checking* [30,12] relates observed behavior to an existing process model, process *enhancement* [2,6] repairs models or extends them e.g., with performance and resource information, and *predictive process monitoring* [22,17] forecasts how process instances may unfold during execution.

The maturity of those techniques has led to an increasing adoption of process mining in industry projects, where analysts often find answers to business problems through a divide-and-conquer strategy by breaking down those problems into fine-grain information needs [10]. Here, process discovery plays a crucial role, as analysts interpret the properties of the discovered models to derive insights [32] that then serve as a foundation for understanding related aspects [1,18]. If interpreted carelessly, process discovery insights can hence negatively affect downstream analysis. Thus, *evaluating* insights

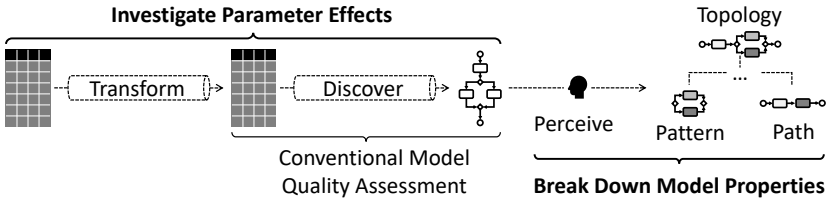


Fig. 1: An extended perspective for the evaluation of process discovery results

from mining, particularly discovery, should be a key activity in each project [10,25] to confirm findings and to turn them into reliable and actionable insights [32]. Besides verifying scripts or tool configurations, consulting domain experts, or investigating the process environment, analysts can also perform *data-driven evaluation* [37].

Commonly, discovery results are evaluated by means of model-centric metrics like fitness, precision, generalization, and simplicity [9,15], which are e.g., computed via conformance checking [12,30] with the log that served as input to the discovery algorithm. Those metrics are valuable for assessing the reliability of discovery algorithms, and we want to complement them by expanding the evaluation perspective, as shown in Figure 1. Analysts typically set up *process discovery pipelines* to transform logs before discovering a model. While necessary to manage log imperfections and complexity, such a pipeline potentially constrains the validity of the behavior covered by the discovered model. Thus, we propose to *examine how pipeline parameters affect properties of the discovered process models* at different granularity levels, because analysts often focus on specific execution paths and patterns to break down the model topology [18].

To this end, we propose a method to investigate the *consistency* of model properties by means of *uncertainty and sensitivity analysis* [36]. Our primary goal is to enable *what-if* analyses in which the reliability of insights is assessed by examining relationships between pipeline parameters and model properties. Yet, the method can also be applied to guide the pipeline definition, or to generate insights from those relationships. In more detail, we present a configurable framework to evaluate, if user-defined model properties are consistent with results from varied configurations of a user-defined pipeline and to quantify the contribution of individual pipeline parameters towards inconsistencies. In doing so, we follow recent work in software engineering [3], which defines a notion of debugging for machine-learning (ML) pipelines. As such, our proposal can be understood as a method for *debugging process discovery pipelines*.

Following, we discuss the problem in Section 2, relying on observations from a competitive process analysis challenge and an illustrative analysis of a moderately complex real-world dataset. We then outline the framework and demonstrate its application using the same dataset in Section 3. In a separate experiment, we investigate our framework in a realistic analysis setting based on another real-world dataset with high complexity in Section 4. Here, we substantiate the utility of our framework by showing that its output is founded in observations by external analysts and theory. The results demonstrate that our debugging framework is a valuable addition to the process mining toolbox: in addition to existing guidelines, patterns, and tools which we discuss in Section 5, it enables analysts and their audiences to comprehend the degree to which properties of discovered models are constrained by analytical decisions in a specific context. Finally, we conclude the paper and discuss future directions in Section 6.

2 Basic Terminology & Problem Illustration

An *event log* L is a set of traces and each trace is an ordered sequence of events. Event logs also contain features that describe properties of events and traces, such as case identifiers, event timestamps, or activity names. A *process model* P is a directed graph where typed and labeled nodes represent activities, gateways, events, etc., whereas edges depict the control flow. Finally, $\mathcal{L}$ and $\mathcal{P}$ denote the universes of event logs and process models, respectively. Note that for the purposes of this paper this basic understanding is sufficient. We hence omit formal definitions which are e.g., presented in [1, Ch. 3 & 5].

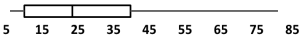
To analyze the process behavior captured in an event log, analysts often define *process discovery pipelines*, either implicitly or explicitly. In this paper, we primarily focus on pipelines that transform a single log into a single model. In the general case, however, a process discovery pipeline can be viewed as a function $\delta : \mathcal{L}^{n_l} \times \mathcal{X}^{n_x} \rightarrow \mathcal{P}^{n_p}$ that takes n_l event logs and a set of n_x *parameters* from the universe of parameters $\mathcal{X}$ and returns n_p process models. Pipelines are assembled by combining transformation and discovery operators. Each operator can be configured via its own set of parameters, all of which are included in the set of parameters that serves as input to the discovery pipeline. Pipelines can be implemented as Python or R-scripts based on packages like dplyr[5], bupaR[6], pandas[7], and pm4py[8], or by incrementally executing tools or components, like ProM plugins[9], but they often involve multiple tools and adhoc scripts [18].

The reasons for analysts to apply discovery pipelines are twofold. On the one hand, logs might contain *imperfections*, such as missing values or outlier behavior. To eliminate those imperfections, analysts filter traces or events, and manipulate features to improve their quality or to enrich logs with data from other information sources. On the other hand, *log complexity* typically poses a challenge in interpreting the data, when logs contain drifts or describe a diverse range of activities or variants. In addition to filtering cases and events, analysts commonly lift the level of abstraction by defining higher level activities or sub-processes and by aggregating the events in the log accordingly. Note that some operations are directly supported by discovery algorithms, e.g., the inductive miner [19] can filter infrequent behavior, while directly-follows graph mining techniques often allow analysts to filter paths and activities based on their frequencies.

In this work, we postulate that the analytical decisions behind the pipeline configuration ultimately constrain the degree to which the behavior depicted in a discovered process model can be generalized. Consider e.g., the following observations from the *business process intelligence challenge* (BPIC), a competition that invites researchers, students, and experts to submit analysis reports for real-world event logs. Table 1 contrasts the complexity of the five event logs from BPIC 2015[10] with the distribution of complexity of the discovered process models presented in the nine submissions. While the event logs are highly complex with 350+ activities and 800+ variants, the majority of the models contains between 6 and 40 activities. We could not reliably quantify the

[5] https://dplyr.tidyverse.org, accessed 2021-05-12
[6] https://www.bupar.net, accessed 2021-05-12
[7] https://pandas.pydata.org, accessed 2021-05-12
[8] https://pm4py.fit.fraunhofer.de, accessed 2021-05-12
[9] http://www.promtools.org/, accessed 2021-05-12
[10] https://www.win.tue.nl/bpi/doku.php?id=2015:challenge, accessed 2021-05-12

Table 1: Complexity of event logs and of discovered models in BPIC 2015

	Logs	1	2	3	4	5
Log Complexity	**# Events**	52,217	44,354	59,681	47,293	59,083
	# Activities	398	410	383	356	389
	# Variants	1,170	828	1,349	1,049	1,153
Model Complexity	**# Activities**					

number of model paths, but observed that the models only allowed for a fraction of the log variants. Moreover, one report in fact included models discovered from the raw logs, to demonstrate that it is impossible to interpret these models. While necessary to manage the cognitive load, the transformations in the underlying pipelines can affect the nature of the discovered model, even if they are less extensive, as illustrated below.

We analyzed the *Sepsis* event log[11] which captures treatments of Sepsis patients in a Dutch hospital [23]. Its complexity is moderate (1,050 cases, 15,214 events, 16 activities), rendering it useful for illustration purposes. We used the default configuration of the inductive miner [19] (infrequent variant, noise `threshold` = 0.2) to discover a process model. But, we first filtered out short cases with an execution duration smaller than `minDuration` based on a common assumption that short cases represent incomplete or outlier behavior. Next, we abstracted the log by aggregating activities related to the release of patients. That is, if `consolidate` is set to `true`, all release-related events are re-labeled and in each trace all but the last release-related events are removed. Note that these transformations are not presented here as the ideal way to handle the log, but merely for illustration purposes. We chose the transformations, as we observed that they were commonly applied in submissions to different editions of the BPIC.

By varying the two parameters, we yielded the four models shown in Figure 2. The differences between the models demonstrate that discovery results can strongly depend on a specific pipeline configuration and hence might be inconsistent with models discovered using varied configurations. For instance, model 1 indicates that the registration activities are executed in arbitrary order before all other activities; in model 2 and 3 they

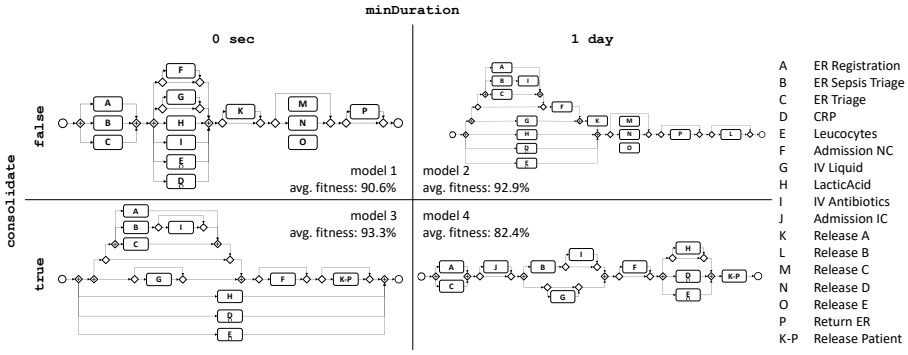


Fig. 2: Sepsis results for different pipeline configurations (fitness calculated with the multi-perspective process explorer in ProM with the transformed event logs).

[11] https://data.4tu.nl/articles/dataset/Sepsis_Cases_-_Event_Log/12707639, accessed 2021-03-12

Table 2: Effects of the analyst's awareness of result uncertainties (adapted from [31])

		Discovery Result	
		No Uncertainties	**Uncertainties**
Analyst	**Aware**	trust in insight: high decision making: unaffected	trust in insight: medium-low decision making: largely unaffected
	Mistaken	trust in insight: medium-low decision making: affected	trust in insight: high decision making: severely affected
	Unaware	trust in insight: medium decision making: unaffected	trust in insight: medium decision making: severely affected

are optional and parallel to the treatment activities; and in model 4 the registration activity B requires the completion of the two remaining registration activities A and C. Differences consequently also exist at the level of the model topology. Yet, the models achieve similar fitness values. This shows that model-centric quality metrics may not reflect how pipeline configurations impact properties of the discovered process models.

In summary, we demonstrated that, while configuring a discovery pipeline is necessary to manage log imperfections and complexity, it might constrain the discovered model, when varied pipeline configurations yield inconsistent outputs. This can ultimately affect the certainty with which insights can be inferred from a discovered model. Following the awareness classification from [31] (see Table 2), we argue that insight uncertainties can impact the decision making that is based on the insights. In the presence of uncertainties, the chance of error due to unjustified trust in the insights is high, when analysts are unaware of or mistakenly assume the absence of uncertainties. But also in the absence of uncertainties, decision making might be impaired when analysts unnecessarily question the insight validity due to mistakenly assuming that uncertainties exist. While in the remaining cases the decision making is usually not affected, analysts (and their audiences) should ideally always be aware of the level of uncertainty that is associated with the insights and of its root causes.

3 Debugging of Process Discovery Pipelines

The necessity to address log imperfections and complexity via pipeline operations can result in uncertain insights and impaired decision making (see Section 2). Such uncertainty can stem from stochastic operators, but most often is introduced by the pipeline parameters. For example, while there might be a plausible range of threshold values for a filter that removes outlier traces with short durations, the precise value can be uncertain. Diagnosing such uncertainty by manually varying parameters and inspecting the respective outputs is infeasible due to the number of configurations needed to obtain reliable conclusions, especially when model and pipeline complexities, or parameter interactions are present. Moreover, it is not transparent to the model audience. Hence, to assist analysts in debugging their discovery pipelines, we pursue two objectives:

O1: *Assess the consistency of model properties* to unveil potential pipeline constraints.
O2: *Quantify the influence of parameters* to provide explanations for inconsistencies.

While our approach could be used to evaluate steps in pipelines generally, we designed it with the purpose of allowing an analyst to achieve objectives O1 and O2 for a concrete

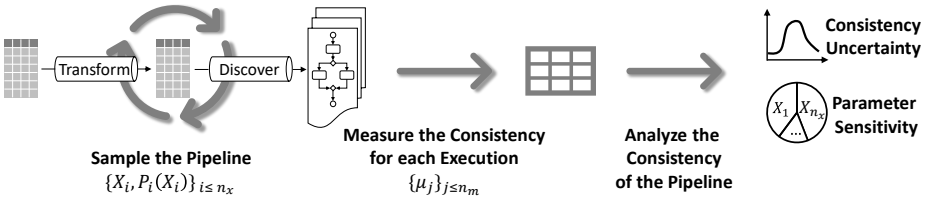


Fig. 3: Framework for investigating property consistency in process discovery pipelines

case. As such, the standard situation for applying our framework is: an analyst has created a concrete pipeline with a concrete parameter configuration to generate a *baseline model*. The analyst then investigates how the parameters influence the model properties (i) to substantiate insights inferred from the baseline model, (ii) to iteratively construct a reliable pipeline, or (iii) to generate insights from parameter / property relationships. In all cases, the metrics are calculated relative to the properties of the baseline model.

To this end, one conceivable strategy is to instrument the pipeline and to track the validity of model properties in all steps [45], i.e., in all intermediate logs and the discovered process models. Yet, as this analysis only considers the current configuration, we would not be able to measure the consistency of model properties with it, or to reason about the general influence of parameters. Hence, we adopt *uncertainty and sensitivity analysis* which provides means to quantify effects of varied pipeline configurations. In this regard, a first option are *one-at-a-time* designs [36, pp. 66–69]. In such a design we would examine both objectives by focusing on each parameter individually. Given a parameter, we would repeatedly change its value and for each value execute the pipeline *without* modifying any of the other parameters. Then, we would use the generated outcomes to examine how variations in the parameter change the pipeline outcome. While this is computationally efficient, the analytical results can be skewed in the presence of parameter interactions [34]. *Global sensitivity analysis* overcomes this limitation by studying the effects of simultaneous parameter changes. Here, *variogram analysis of response surfaces* (VARS) [29] aims to reveal the spatial structure and variability of model outputs. Essentially, VARS models the output space as a variogram function that describes the degree to which model outcomes for a specific parameter configuration X depend on outcomes produced by configurations in the vicinity of X. This variogram function is then used to examine properties of input-output relationships. However, VARS does not provide clear indications for the importance of inputs and thus, they should be used to complement *variance-based sensitivity analysis* [28]. We follow this argumentation and build our framework on the scheme for variance-based sensitivity analysis from [35].

As shown in Figure 3, we first *sample the pipeline* (Section 3.1). That is, we execute the user-defined pipeline $\delta : \mathcal{L}^{n_l} \times \mathcal{X}^{n_x} \rightarrow \mathcal{P}^{n_p}$ multiple times to generate process models for different parameter configurations. Here, we consider event logs to be constants. This effectively turns discovery pipelines into functions $\delta_X : \mathcal{X}^{n_x} \rightarrow \mathcal{P}^{n_p}$ that only take parameters as input. To guide the exploration and the parameter sampling, analysts must specify the *relevant* parameters and their probability measures $\{(X_i, P_i(X_i))\}_{i \leq n_x}$. Next, we *measure the property consistency for each execution* (Section 3.2), requiring the analysts to manually determine the model properties for which they want to measure the *consistency*, i.e., the degree to which a (set of) model(s) produced in a

single execution satisfies this property. In particular, the analyst must provide a set of n_m *property consistency measurements* $\{\mu_j\}_{j \leq n_m}$ where each function $\mu_j : \mathcal{P}^{n'_{p,j}} \rightarrow [0,1]$ represents a specific property and returns the consistency for this property as observed in a set of $n'_{p,j}$ process models: a value of 0 indicates total inconsistency, a value of 1 perfect consistency, and values in between degrees of consistency. Lastly, we *analyze the property consistency of the pipeline* (Section 3.3): an uncertainty analysis assesses the degree to which a model property changes when pipeline parameters vary (O1), whereas sensitivity analysis quantifies the contribution of individual parameters to potential inconsistencies (O2). Below, we describe each step using the Sepsis experiment from Section 2 for illustration purposes.

3.1 Sampling the Pipeline

To explore the output of different pipeline configurations, we first create a $k \times n_x$ configuration matrix $\mathbf{A}$ which comprises the configurations for k pipeline executions. Each configuration contains n_x values, one per relevant parameter X_i. We use the configurations in $\mathbf{A}$ to assess whether the pipeline yields inconsistencies (O1, see Section 3.3). If there are inconsistencies and it must be analyzed how parameters contribute to them (O2, see Section 3.3), then for each parameter X_i we create an additional $k \times n_x$ configuration matrix $\mathbf{AB}_i$ by copying $\mathbf{A}$ and varying the values in the ith column which defines the values for parameter X_i. Comparing the results obtained from the configurations in $\mathbf{A}$ and $\mathbf{AB}_i$ allows us to quantify the influence of parameter X_i. Thus, when desired, O2 requires $k \times n_x$ additional pipeline executions, yielding a total of $k \times (n_x + 1)$ executions.

For a reliable analysis we need configurations that (i) sufficiently sample the entire parameter space and (ii) systematically vary the parameter values. We achieve this based on the procedure that yielded the best results in a comparative evaluation by Saltelli et al. [35]. First, we use a *low-discrepancy* sequence to generate two temporary $k \times n_x$ matrices $\mathbf{A}^t$ and $\mathbf{B}^t$ where each row is a point in the n_x-dimensional unit cube. Low-discrepancy sequences ensure that the parameter space is evenly sampled. We here use the *Sobol' sequence* [39] which, in contrast to sequences like the Latin Hypercube design, has the advantage that we do not necessarily need to fix the sample size, but could in principle dynamically generate new configurations until the analysis results converge. We use the Sobol' sequence to generate a $k \times 2n_x$ matrix that is split in half to obtain the temporary matrices $\mathbf{A}^t$ and $\mathbf{B}^t$ from the left and right half, respectively. While we derive $\mathbf{A}$ directly from $\mathbf{A}^t$, we use $\mathbf{B}^t$ to create the temporary matrices $\{\mathbf{AB}^t_i\}_{i \leq n_x}$ using the *radial sampling* strategy [33]. That is, for each parameter X_i we construct $\mathbf{AB}^t_i$ by copying $\mathbf{A}^t$ and replacing the i-th column with the respective column from $\mathbf{B}^t$. Lastly, we obtain the configuration matrices ($\mathbf{A}$ and $\{\mathbf{AB}_i\}_{i \leq n_x}$) by interpreting the values in the temporary matrices as probabilities: for each parameter we convert each value p in the i-th columns of the temporary matrices to a parameter value x for X_i so that the respective cumulative probability yields the probability p for value x, i.e., $P_i(X_i \leq x) = p$. The final step is to execute the discovery pipeline for each configuration in $\mathbf{A}$ to discover the process models. The configurations from $\{\mathbf{AB_i}\}_{i \leq n_x}$ are only executed, if inconsistencies exist for which the analyst wishes to inspect the influence of parameters.

In our running example, the Sepsis experiment, we sample the pipeline for the parameters `minDuration`, `consolidate`, and `threshold`, in this order of parameters.

We here also consider the `threshold` parameter, because in Section 2 it was set to 0.2 by default and might have influenced the results. For `consolidate` and `threshold` we use uniform distributions over their entire domains ({`false`,`true`} and [0, 1]), whereas for `minDuration` we use the empirical distribution of case durations in the log for all values ≤ 2 days. Setting `minDuration` to 2 days would exclude about 29% of the cases, and hence we chose this value as an upper bound. Taking a concrete example for a configuration, say the current configuration from $\mathbf{A}^t$ or $\mathbf{AB}_i^t$ is (0.7, 0.6, 0.3); then our approach derives the following parameter values as per the above use of the cumulative probabilities. The 70^{th} percentile of the actual data for `minDuration` is at 4h 10min, and therefore we get `minDuration = 4h 10min`. 0.6 > 0.5, hence we get `consolidate=true`. For `threshold`, the uniform distribution equals the identity function, hence `threshold`=0.3. We set the sample size k to 1,000 resulting in 1,000 executions for O1 and (3 × 1,000) = 3,000 executions for O2.

3.2 Measuring the Property Consistency for a Single Execution

Within our framework, analysts can investigate the consistency of the model topology and of fine-grained model properties like execution patterns and paths by defining property consistency measurements $\mu : \mathcal{P}^{n'_P} \rightarrow [0, 1]$. While analysts can provide any measurement, we propose two specific measurements for single models ($n'_P = 1$). Both functions rely on the *causal behavioral profile* [42] which captures behavioral relations between a set of activities T as observed in a set of executions E. The *causal behavioral profile* is defined as $C_{T,E} = \{\rightsquigarrow, +, \|, \gg\}$ where activity pairs $(t_1, t_2) \in T \times T$ are

1. in *strict order* ($t_1 \rightsquigarrow t_2$), if in all executions with t_1 and t_2, t_1 occurs before t_2;
2. in *interleaving order* ($t_1 \parallel t_2$), if they can be executed in arbitrary order;
3. *exclusive* ($t_1 + t_2$), if they are never part of the same execution; and
4. *co-occurring* ($t_1 \gg t_2$), if the presence of t_1 implies the presence of t_2.

We chose behavioral profiles as a foundation for the concrete consistency measurements, as they have been applied for various tasks including process monitoring, complex event processing, conformance checking, and most importantly model consistency assessment [43]. Moreover, they can be computed from heterogeneous inputs. Considering that each trace represents an execution, they can straightforwardly be derived from logs. An efficient computation for *sound* process models [42] derives the profile from a tree representation of the process model. This computation can easily be adopted for discovery algorithms that output process trees such as the inductive miner [19]. For directly-follows graphs with a dedicated start and a dedicated end node, every path from start to end is an execution. Besides these beneficial properties, behavioral profiles might however inaccurately represent behavioral relationships in some cases [27]. Hence, a comparative evaluation of consistency measures is required in future work.

The first type of measurement is the *profile-based consistency* $\mu_C : P \rightarrow [0, 1]$. It requires the provision of a base profile C_{T_b,E_b}. Then, it applies the *degree of consistency* metric from [41] to compute a consistency score for C_{T_b,E_b} and a profile C_{T_d,E_d} derived from a discovered process model. This metric relies on an *alignment* of the activities from T_b and T_d. It hence allows us to compare profiles at the same and at different levels of granularity. If two profiles are at the same level of granularity, all activities with

equal labels are aligned. Otherwise, the pipeline includes a log abstraction step in which fine-grained activities are mapped to higher-level activities e.g., using manually defined hierarchies or automated label comparison [16]. This mapping defines the alignment. Based on the alignment, the first step is to determine the sets of aligned activities T_b^a and T_d^a which contain all activities from T_b and T_d for which the other activity set contains aligned activities. The metric then determines the count γ of activity pairs in $T_b^a \times T_b^a$ and $T_d^a \times T_d^a$ whose relations defined by C_{T_b,E_b} and C_{T_d,E_d} match the relations of the aligned activity pairs from the other profile. The relations of two aligned activity pairs $(t_b', t_b'') \in T_b^a \times T_b^a$ and $(t_d', t_d'') \in T_d^a \times T_d^a$ match, if both pairs are in strict order, interleaving order or exclusive, and they either co-occur or not. If an activity pair (t', t'') is aligned with multiple pairs, then the relations of all these pairs must match the relations of (t', t''). Finally, γ is divided by the number of aligned activity pairs $|T_b^a \times T_b^a| + |T_d^a \times T_d^a|$. In this work, we primarily use the profile from the baseline model discovered with a specific pipeline configuration to track the degree to which behavioral relations change when parameters change. Similar to model-centric quality metrics [9], it is also conceivable to check, if the discovered model accurately reflects the relations in a log, potentially produced during pipeline execution.

A break down of the model topology to investigate more fine-grain aspects can be achieved by removing activities from the base profile to focus on certain activity sets. Additionally, the *rule-based consistency* $\mu_R : P \to \{0, 1\}$ enables analysts to specify arbitrary *rules* in terms of boolean expressions which define relations that need to hold between specific activities, e.g., that an activity α must be in strict order with an activity β. The function then returns a value of 1, if the profile derived from the discovered model adheres to the rule and a value of 0 otherwise. Note that this is similar to the use of declarative rules which are defined at the level of events and traces, whereas the rule-based consistency relies on the more abstract level of the behavioral profile.

In the Sepsis example, we observed some inconsistencies at the model and at the activity level. Here, we focus on three properties for which we analyze the pipeline consistency below in Section 3.3. First, we use the profile-based consistency to evaluate the model that we obtained, when setting `minDuration` to 2 days, `consolidate` to `true`, and `threshold` to 0.2 (I1), see lower right corner of Figure 2. Additionally, we use the rule-based consistency to diagnose specific inconsistencies that we observed when varying the parameters in Figure 2. In particular, we check if the registration activities *A* and *C* occur before all other activities (I2), and if the release activities generally occur at the end of the process (I3). Note that we evaluate all three consistencies based on the same set of configurations and discovered process models, respectively.

3.3 Analyzing the Property Consistency for the Pipeline

The last step conducts the analyses postulated by the two objectives. We first address O1 and examine the uncertainty associated with model properties based on the provided consistency measurements $\{\mu_j\}_{j \leq n_m}$. To this end, we compose the discovery pipeline $\delta_X : \mathcal{X}^{n_x} \to \mathcal{P}^{n_p}$ and each consistency measurement $\mu_j : \mathcal{P}^{n_{p,j}} \to [0, 1]$ to functions $f_j = \mu_j \circ \delta_X$ that measure the property consistency for models produced by a given pipeline configuration. This requires that the consistency functions take as many process models as input as discovered by the pipeline in a single execution, i.e., $n_p = n_{p,j}$.

For a measurement μ_j, we first calculate the *mean consistency* $\overline{f_j} = \frac{1}{k}\sum_{l=1}^{k} f_j(\mathbf{A})_l$ over all configurations from the configuration matrix $\mathbf{A}$ (see Section 3.1). If the mean consistency is equal or very close to 1 (or 0 respectively), we know that the respective property is (not) free of constrains and hence generally (in-)valid. In all other cases, there is uncertainty regarding the conditions that cause inconsistencies and we next estimate the *consistency variance* $\hat{V}(f_j) = \frac{1}{k}\sum_{l=1}^{k}\left(f_j(\mathbf{A})_l^2 - \overline{\mu_j}^2\right)$. If the variance is close to 0, we can infer that all pipeline configurations yield similar consistency values and that there likely is a systematic difference between the property from the baseline model and the properties of the pipeline output, generally. Such a difference can be explored by comparing the originally discovered model to a few models generated with different configurations. Here, the analyst can also resort to restricting the base profile or defining rule-based consistencies, in order to investigate differences at a more fine-grained level.

Larger variance values indicate that varied pipeline configurations yield process models with different levels of consistency. To analyze the influence of parameters as per O2, we compute the *total effect index* $S_{i,j}$ for each parameter X_i [13]. It measures the contribution of parameter X_i to the variance in the consistency measurement μ_j and considers all variance that is directly caused by X_i and by interactions with other parameters. As suggested in [35], we here use the estimator from [14]: $\hat{S}_{i,j} = \frac{1}{2k\cdot\hat{V}(f_j)}\sum_{l=1}^{k}\left(f_j(\mathbf{A})_l - f_j(\mathbf{AB}_i)_l\right)^2$. This estimator relies on the results of the configuration matrix $\mathbf{AB}_i$. The higher the value of the index for a parameter, the more it contributes to the variance in the consistency measurement. If the sum of the indexes is larger than 1 $\left(\sum_{i=1}^{n_x}(\hat{S}_{i,j}) > 1\right)$ the parameters definitely interact.

We conclude by analyzing the pipeline consistency for the Sepsis experiment considering the sampling configuration and properties from Section 3.1 and 3.2. The mean model consistency (I1) is $\overline{f_1} = .57$ and for the two rule-based measures (I2, I3) we yield mean consistencies of $\overline{f_2} = .08$ and $\overline{f_3} = .21$. These low values are in line with our observations from Figure 2, because they indicate that the behavioral relations in the baseline model are associated with uncertainty, especially the relations of the registration and release activities. The variances ($\hat{V}(f_1) = .06$, $\hat{V}(f_2) = .07$, $\hat{V}(f_3) = .16$) point to non-systematic differences which are attributed to all parameters. That is because all consistency / parameter combinations yield high total effect indexes on the interval [.71, .92]. This implies that the handling of the log is not optimal and should be changed, not least because the indexes reveal that there is significant parameter interaction.

4 Experiment

The primary objective of our experiment is to study whether the framework provides a reliable foundation for investigating the effects of discovery pipeline operations on the discovered model and its properties. In the following, we first outline and justify our experimental design in Section 4.1. After that, we discuss our results in Section 4.2.

4.1 Experimental Design

Uncertainty and sensitivity analysis are mature techniques that have been studied intensively, e.g., in [13,14,28,29,33,35,36], and hence provide a solid foundation for our

work. Software engineering for machine learning [3] is an emerging topic, and has not yet been adopted for process mining (see Section 5). Hence, we validate our framework using a *single-case mechanism experiment*, a suitable method for investigating the application of existing technology to a new phenomenon [44, Ch. 18]. To mitigate the effects of a limited *external validity* associated with such a design, i.e., the degree to which the findings can be generalized, we attached great importance to strengthening the *ecological validity*, i.e., the realism with which the setup resembles real-world circumstances, and to minimizing the threat of *experimenter bias*. Moreover, to ensure transparency and reproducibility, we followed open science principles by relying on public data and by publishing our source code[12]. In more detail, we decided to use the BPIC 2015 dataset from Section 2. It is a highly complex (see Table 1), publicly available, real-world dataset for which nine independent analysis reports were published. The latter allows us to setup a representative discovery pipeline based on operations commonly applied by external parties on this dataset. We merely use the reports to guide the pipeline setup. It is not our intention to judge the analysts' practices, for which an exact replication of a pipeline would be required (which is neither desired nor feasible with the level of detail in the reports). The dataset contains five event logs from applications for building permits in different Dutch municipalities. Hence, we can reuse the sample pipeline to analyze our framework in (slightly) varied circumstances.

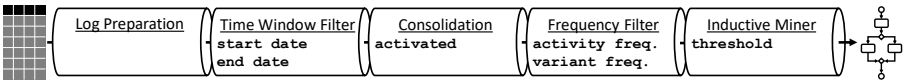
We first categorized the applied transformation operations from the reports and assembled the three most common operations into the pipeline from the last row of Table 3. First, the *log preparation* loads the log and performs computations that ease the analysis. That is, the log specifies an activity code which is the activity identifier, but also contains a sub-process identifier and an order index. As the sub-process identifier is used for log consolidation, we extract it into a separate feature. Because events were logged in batch with overlapping timestamps, we follow advice from the BPIC organizers and establish the execution order based on the order index. After that, we apply a *time window filter* to remove traces that started or completed outside a window defined by pipeline parameters `start` and `end date`. This operation addresses the drifts in the log which impact the discovery, and we here consider a time window from summer 2013 to spring 2014 in which no drift occurred. If parameter `activated` is set to `true`, we perform a *consolidation* in which we define the sub-process identifier as the activity classifier. Further, in each trace we only keep the first and last sub-process event and set the event lifecycle state to `started` for the first event, and retain `completed` for the last. Next, a *frequency filter* can reduce the complexity of the discovered process model by selecting events and traces based on the `activity` and `variant frequency`. Lastly, we apply the infrequent lifecycle variant of the inductive miner [19] where the noise `threshold` also allows for filtering behavior.

To systematically study the effects of combining different operations, we vary the subset of relevant parameters from the above six parameters, and set the remaining parameters to default values. The relevance of parameters for the variants and their probability measures are summarized in Table 3. V1 establishes a baseline in which we only vary the parameters of the time window filter. Here, we expect that the absence of drifts in the considered period (summer 2013 to spring 2014) guarantees a consistent

[12] https://bitbucket.csiro.au/users/kli039/repos/bpm-2021-debugging-experiments

Table 3: Pipeline specification for the experiment including the parameters' *emp*-irical or *uni*-form distributions; their *rel*-evance for the variants V1–V5 where a default value is provided for irrelevant parameters (0, 1, f – false, t – true); and the parameter values that were used to generate baseline profiles for different consistency measurements.

	Probabilities			Variants					Baseline Profile Generation				
Parameter	Type	From	To	V1	V2	V3	V4	V5	norm	abst	simp	mod	comp
start date	emp	1/5/13	30/6/13	always relevant					always set to 1/6/13				
end date	emp	1/4/13	31/5/14	always relevant					always set to 30/4/14				
activated	uni	f	t	f	rel	f	f	rel	f	t	f	f	f
activity freq.	uni	0	1	1	1	rel	rel	rel	1	1	.2	.35	.5
variant freq.	uni	0	1	1	1	rel	1	rel	always 1				
threshold	uni	0	1	0	0	0	rel	rel	always 0				



discovery for slightly varied `start` and `end dates`. To study the impact of model consolidation, V2 additionally considers the `activated` parameters. Here, we expect that the information loss which is inadvertently linked to abstraction leads to a drop in the consistency, but that the discovered models are largely consistent, as we rely on a clearly defined process hierarchy. In V3 and V4, we add different ways of behavior filtering to V1: while both variants utilize the `activity frequency`, V3 additionally combines it with the `variant frequency` and V4 with the noise `threshold`. We hypothesize that these filters interact with the time window filter, which influences the frequencies in the intermediate log. Finally, in V5 all parameters are relevant.

To investigate the pipeline consistency, we focus on the overall model consistency using the profile-based consistency. In this regard, different baseline models and thus base profiles emulate different degrees of complexity of discovered models (see Table 3). All profiles are derived from the log for the default time window. The *normative* (norm) profile has the highest complexity. It is discovered directly from the default time window log and used for all variants. For V2, we also use an *abstract* (abst) profile obtained by activating the consolidation. Lastly, for V3 and V4 we aimed to replicate different model complexities in line with the model complexities found in the reports (Table 1). We generate the *simple* (simp), *moderate* (mod), and *complex* (comp) profiles by varying the `activity frequency` to obtain models with ≈10, ≈20 and ≈35 of the most frequent activities. We did not use the `variant frequency` or noise `threshold`, as their effects on the model complexity differed across the five logs. Yet, the profile-based consistency still allows us to assess their influence on the discovery results.

4.2 Results

In the analysis, we considered a sample size of $k = 1{,}000$ for all combinations of pipeline variants and consistency measurements. To ensure that this sample size yields reliable results, we first investigated the convergence of the mean consistencies, variances, and total effect indexes. That is, we computed the values that we obtain for these

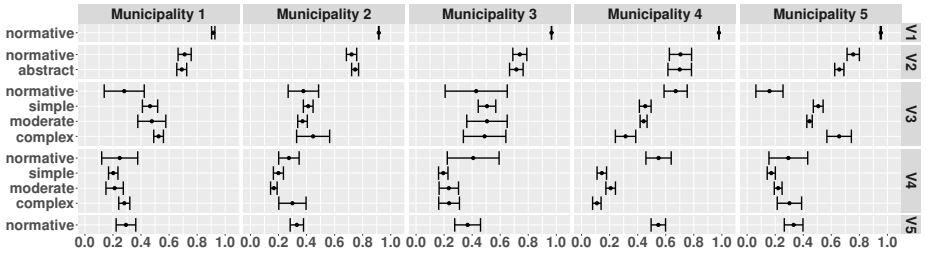


Fig. 4: Mean consistencies (dot) and variances (error bars) for pipeline variants

measures for sample sizes less than 1,000 and observed that for sample sizes larger than 500, all measures yield values that are very close to the respective values obtained for $k = 1{,}000$ on all five logs for all variant/measurement combinations. While this ensures the reliability of our experiment, it also demonstrates that measuring the convergence of the values is a strategy to control the number of pipeline executions in real-world situations. We did not investigate the run-time performance explicitly, but observed that the inductive miner accounted for a large part of the execution time and that its performance depended (unsurprisingly) on the complexity of the input log. To compute all metrics per variant and dataset, on a customary laptop (Processor: i5-8350U 1.70 GHz; RAM: 16GB) and using parallel execution we yielded execution times between one and two hours for (V1); but below 5min for V3–V5, due to complexity reductions in the intermediary logs. Note that this is only a rough indication for the run-time performance, for which we leave deeper investigation and optimization to future work.

We first investigate the uncertainty for each variant and consistency combination, see Figure 4. A first observation is that the consistency of the normative model is very high ($\overline{f_j} > .9$) for V1. This is in line with our expectations, as we knew from the reports that the considered period does not contain drifts. Slight variations in the model can be attributed to a few outlier cases that might occur around the default start and end date. For V2 we also confirm our expectations, as the model consistency drops ($\overline{f_j} > .7$) due to some information loss caused by the consolidation, but is still high. Note that this holds for the abstracted and the normative model, indicating that log abstraction is a reliable means for complexity management. Lastly, the variants that apply filtering (V3–V5) yield very low consistency measures ($\overline{f_j} < .5$). While we expected some interaction with other parameters, we were surprised by the magnitude of the effect of this interaction. However, this observation is in line with guidelines from [11] that postulate to carefully apply random subset selection, as it – in contrast to strategic selection, like the date window filter – can affect the quality of the discovered model. We consider the filter parameters from V3–V5 to fall in this category, as it is hard for analysts to predict the effects of certain value combinations. Moreover, the negative effects pertain all base profiles which shows that the filters affect a large range of the relations and that a broad range of possible behavior can be generated by modifying the respective parameters. Overall, the coherence of our expectations and existing guidelines with the experiment results substantiates the reliability of the consistency measurement.

To study the sensitivity analysis, we focused on the three variants with filtering (V3–V5) and the normative base profile which overall yielded the largest variance across all logs. The total effect indexes for all parameters per variant and log are shown in Figure 5

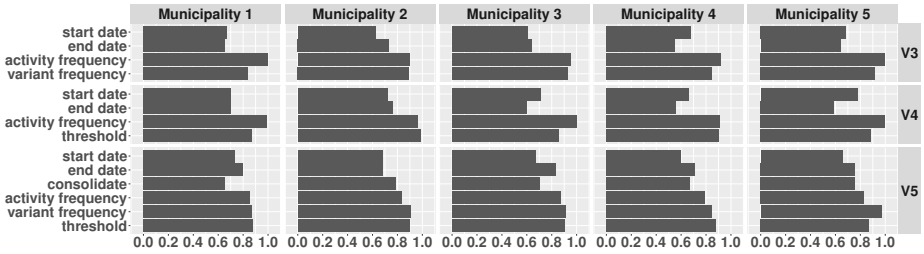


Fig. 5: Total effect indexes for the normative profile and variants V3–V5

where higher values for a parameter indicate a stronger contribution of this parameter to the variance. In line with the uncertainty analysis for the variants, the total effect indexes show the `frequency` and `threshold` parameters to contribute the most to the uncertainty in the model topology. This provides evidence towards the utility of the sensitivity analysis: an analyst can determine the most influential parameters without manually inspecting possible parameter or pipeline variations. Another interesting finding is that the time window filter and consolidation parameters, which without filtering only impacted the consistency a bit, have a stronger influence in variants V3–V5. This demonstrates that analysts need to carefully assemble discovery pipelines and cannot assume that a 'stable' operation can be straightforwardly reused in other contexts.

5 Related Work

Research has studied issues related to data quality and quantity, in order to ensure that high quality process models can be obtained from event logs. Classifications of data quality issues [8] and data quality patterns for event logs [40] allow for systematic cleaning of event logs to increase process mining result quality. Fitness, precision, generalization, and simplicity have been adopted as metrics to evaluate the quality of a process model based on the event log that served as the input for a process discovery algorithm [1]. Conformance checking allows to obtain further details about if and how an event log deviates from a process model for qualitative evaluation [12,30]. Also, methods have been proposed to balance the behavioral quality of a discovered process model with its complexity, in order to facilitate human inspection. For example, in [20] event attributes are used to generate hierarchical process models that better represent different levels of process granularity. A statistical pre-processing framework for event logs that reduces the amount of data needed to produce high quality process models is presented in [7]. Similarly, the influence of subset selection on the model quality was examined in [11] where it was shown that, in contrast to random-based selection, strategic subset selection increases the model quality. The taxonomy of log and model uncertainty from [26] considers issues like incorrectness, coarseness, and ambiguity, and allows for obtaining upper and lower uncertainty bounds for conformance checking.

Related work also proposed approaches for automatically extracting and evaluating process discovery insights. An automatic approach that compares different process variants with the goal to obtain valuable insights is introduced in [6]. In more detail, the best and worst-performing variants with respect to a set of key performance indicators are

determined and their differences are presented to the analyst. ProcessExplorer [38] automatically computes potential subsets of cases and evaluates the interestingness based on statistical differences between insights from the subsets and from the entire event log. Leemans et al. [21] introduce an automatic extraction approach to obtain cohorts from event logs via trace attribute analysis. The authors measure the stochastic distance between trace attribute cohorts to identify their influence to the process model behavior.

Complementary to these techniques, patterns, and guidelines, our consistency framework enables analysts to, in a concrete context, explicate how their decisions, that underlie the configuration of a discovery pipeline including its log transformations and discovery algorithms, affect model properties at different granularity levels.

6 Conclusion

In this work we presented a first framework for debugging of process discovery pipelines. We demonstrated the potential effects of pipeline operations on the discovered models and discussed the implications for downstream decision making. Next, we proposed a debugging framework which relies on uncertainty and sensitivity analysis, in order to assist analysts in assessing the consistency of their insights and to quantify the contribution of pipeline parameters to potential inconsistencies. In an experiment on real-world event logs, we assessed the utility of our framework and found that the uncertainties and explanations delivered by the framework were well-grounded.

As mentioned in Section 3.2, comparative evaluations of consistency measures are required to improve the framework's applicability. Beyond that, research opportunities ensue specifically regarding its *usability*, *computational performance*, and *broader application and evaluation*. Usability topics comprise suitable user interfaces for tools, but also the generalization towards other process mining methods including declarative process mining; support for determining relevant parameters (e.g., via screening [36]) and their probability distributions; and means to diagnose and break down inconsistencies. Moreover, repeatedly executing a pipeline for different configurations can be time-consuming. While screening methods can help to reduce the number of relevant parameters, integrated uncertainty propagation [24] or emulators [36] might speed up the analysis. Lastly, applying the framework to a larger set of real-world scenarios could potentially reveal and confirm (anti-)patterns for process mining pipelines [40].

In general, we believe that applying software engineering practices, as proposed in the context of machine learning [3], is relevant for process mining as well. While traditionally process mining techniques have been made available via visual idioms which combine visual representations and user interaction techniques, packages like BupaR and pm4py have brought process mining to open data processing environments like R, Python, Apache Spark, etc. This enables a paradigm shift towards script-based analysis, where the ability to seamlessly integrate data processing, data mining, and machine learning techniques and tools can ease the definition, execution, documentation, and sharing of process mining pipelines, and reduce their fragmentation. In this regard, challenges from machine learning include testing, experiment management, transparency, and troubleshooting [4]. Empirical studies into the practices of process analysts, such as [18], can help to refine those challenges in the context of process mining.

References

1. van der Aalst, W.: Process Mining: Data Science in Action. Springer (2016)
2. Adriansyah, A., Buijs, J.C.A.M.: Mining process performance from event logs. In: BPM Workshops. pp. 217–218 (2013)
3. Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., Zimmermann, T.: Software engineering for machine learning: A case study. In: ICSE SEIP. p. 291–300 (2019)
4. Arpteg, A., Brinne, B., Crnkovic-Friis, L., Bosch, J.: Software engineering challenges of deep learning. In: SEAA. pp. 50–59 (2018)
5. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Polyvyanyy, A.: Split miner: automated discovery of accurate and simple business process models from event logs. Knowl Inf Syst **59**, 251–284 (2019)
6. Ballambettu, N.P., Suresh, M.A., Bose, R.P.J.C.: Analyzing process variants to understand differences in key performance indices. In: CAISE. pp. 298–313 (2017)
7. Bauer, M., Senderovich, A., Gal, A., Grunske, L., Weidlich, M.: How much event data is enough? a statistical framework for process discovery. In: CAISE. pp. 239–256 (2018)
8. Bose, R.P.J.C., Mans, R.S., Van Der Aalst, W.M.P.: Wanna improve process mining results? In: IEEE SSCI. pp. 127–134 (2013)
9. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. Int J Coop Inf Syst **23**(01), 1440001 (2014)
10. van Eck, M.L., Lu, X., Leemans, S.J.J., van der Aalst, W.M.P.: PM^2: A process mining project methodology. In: CAISE. pp. 297–313 (2015)
11. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: The Impact of Event Log Subset Selection on the Performance of Process Discovery Algorithms. In: ADBIS. pp. 391–404 (2019)
12. García-Bañuelos, L., van Beest, N.R.T.P., Dumas, M., Rosa, M.L., Mertens, W.: Complete and interpretable conformance checking of business processes. IEEE Trans Softw Eng **44**(3), 262–290 (2018)
13. Homma, T., Saltelli, A.: Importance measures in global sensitivity analysis of nonlinear models. Reliab Eng Syst Saf **52**(1), 1–17 (1996)
14. Jansen, M.J.W.: Analysis of variance designs for model output. Comput Phys Commun **117**(1), 35–43 (1999)
15. Kalenkova, A., Polyvyanyy, A., La Rosa, M.: A framework for estimating simplicity of automatically discovered process models based on structural and behavioral characteristics. In: BPM. pp. 129–146 (2020)
16. Klinkmüller, C., Weber, I.: Every apprentice needs a master: Feedback-based effectiveness improvements for process model matching. Inf Syst **95**, 101612 (2021)
17. Klinkmüller, C., van Beest, N.R.T.P., Weber, I.: Towards reliable predictive process monitoring. In: CAISE Forum. pp. 163–181 (2018)
18. Klinkmüller, C., Müller, R., Weber, I.: Mining process mining practices: An exploratory characterization of information needs in process analytics. In: BPM. pp. 322–337 (2019)
19. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Petri Nets. pp. 311–329 (2013)
20. Leemans, S.J.J., Goel, K., Van Zelst, S.J.: Using multi-level information in hierarchical process mining: Balancing behavioural quality and model complexity. In: ICPM. pp. 137–144 (2020)
21. Leemans, S.J.J., Shabaninejad, S., Goel, K., Khosravi, H., Sadiq, S., Wynn, M.T.: Identifying Cohorts: Recommending Drill-Downs Based on Differences in Behaviour for Process Mining. In: ER. pp. 92–102 (2020)

22. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: CAISE. pp. 457–472 (2014)
23. Mannhardt, F., Blinde, D.: Analyzing the trajectories of patients with sepsis using process mining. In: BPMDS. pp. 72–80 (2017)
24. Manousakis, I., Goiri, I.n., Bianchini, R., Rigo, S., Nguyen, T.D.: Uncertainty propagation in data processing systems. In: ACM SOOC. pp. 95–106 (2018)
25. Mariscal, G., Marbán, s., Fernández, C.: A survey of data mining and knowledge discovery process models and methodologies. Knowl Eng Rev **25**(2), 137–166 (2010)
26. Pegoraro, M., van der Aalst, W.M.P.: Mining uncertain event data in process mining. In: ICPM. pp. 89–96 (2019)
27. Polyvyanyy, A., Armas-Cervantes, A., Dumas, M., García-Bañuelos, L.: On the expressive power of behavioral profiles. Formal Aspects of Computing **28**(4), 597–613 (2016)
28. Puy, A., Lo Piano, S., Saltelli, A.: Is vars more intuitive and efficient than sobol' indices? Environ Model Softw **137**, 104960 (2021)
29. Razavi, S., Gupta, H.V.: A new framework for comprehensive, robust, and efficient global sensitivity analysis: 1. theory. Water Resour Res **52**(1), 423–439 (2016)
30. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf Syst **33**(1), 64–95 (2008)
31. Sacha, D., Senaratne, H., Kwon, B.C., Ellis, G., Keim, D.A.: The role of uncertainty, awareness, and trust in visual analytics. IEEE Trans Vis Comput Graph **22**(1), 240–249 (2016)
32. Sacha, D., Stoffel, A., Stoffel, F., Kwon, B.C., Ellis, G., Keim, D.A.: Knowledge generation model for visual analytics. IEEE Trans Vis Comput Graph **20**(12), 1604–1613 (2014)
33. Saltelli, A.: Making best use of model evaluations to compute sensitivity indices. Comput Phys Commun **145**(2), 280–297 (2002)
34. Saltelli, A., Aleksankina, K., Becker, W., Fennell, P., Ferretti, F., Holst, N., Li, S., Wu, Q.: Why so many published sensitivity analyses are false: A systematic review of sensitivity analysis practices. Environ Model Softw **114**, 29–39 (2019)
35. Saltelli, A., Annoni, P., Azzini, I., Campolongo, F., Ratto, M., Tarantola, S.: Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index. Comput Phys Commun **181**(2), 259–270 (2010)
36. Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., Tarantola, S.: Global Sensitivity Analysis. The Primer. Wiley (2008)
37. Sargent, R.G.: Verification and validation of simulation models. J Simul pp. 12–24 (2013)
38. Seeliger, A., Sánchez Guinea, A., Nolle, T., Mühlhäuser, M.: Processexplorer: Intelligent process mining guidance. In: BPM (2019)
39. Sobol, I.M.: Uniformly distributed sequences with an additional uniform property. USSR Comput Math Math Phys **16**(5), 236–242 (1976)
40. Suriadi, S., Andrews, R., ter Hofstede, A.H.M., Wynn, M.T.: Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Inf Syst **64**, 132–150 (2017)
41. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Trans Softw Eng **37**(3), 410–429 (2011)
42. Weidlich, M., Polyvyanyy, A., Mendling, J., Weske, M.: Efficient computation of causal behavioural profiles using structural decomposition. In: Petri Nets. pp. 63–83 (2010)
43. Weidlich, M., Polyvyanyy, A., Mendling, J., Weske, M.: Causal behavioural profiles - efficient computation, applications, and evaluation. Fundam Inf **113**(3–4), 399–435 (2011)
44. Wieringa, R.J.: Design Science Methodology for Information Systems and Software Engineering. Springer (2014)
45. Yang, K., Huang, B., Stoyanovich, J., Schelter, S.: Fairness-aware instrumentation of preprocessing pipelines for machine learning. In: HILDA (2020)