

Towards Confirmatory Process Discovery: Making Assertions About the Underlying System

Gert Janssenswillen · Benoît Depaire

Received: 10 October 2017 / Accepted: 21 August 2018 / Published online: 4 December 2018
© Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018

Abstract The focus in the field of process mining, and process discovery in particular, has thus far been on exploring and describing event data by the means of models. Since the obtained models are often directly based on a sample of event data, the question whether they also apply to the real process typically remains unanswered. As the underlying process is unknown in real life, there is a need for unbiased estimators to assess the system-quality of a discovered model, and subsequently make assertions about the process. In this paper, an experiment is described and discussed to analyze whether existing fitness, precision and generalization metrics can be used as unbiased estimators of system fitness and system precision. The results show that important biases exist, which makes it currently nearly impossible to objectively measure the ability of a model to represent the system.

Keywords Process mining · Process discovery · Process quality · Fitness · Precision · Generalization · Exploratory data analysis · Confirmatory data analysis

Accepted after three revisions by Jan Mendling.

G. Janssenswillen · B. Depaire
Hasselt University, Agoralaan Bldg D, 3590 Diepenbeek,
Belgium
e-mail: benoit.depaire@uhasselt.be

G. Janssenswillen (✉)
Research Foundation Flanders (FWO), Egmontstraat 5,
1060 Brussels, Belgium
e-mail: gert.janssenswillen@uhasselt.be

1 Introduction

Nowadays, organizations are storing huge amounts of data related to various business processes. Process mining provides different methods and techniques to analyze and improve these processes, allowing companies to gain a competitive advantage. Initiated with the discovery of work-flow models from event data (Agrawal et al. 1998; Cook and Wolf 1995; Datta 1998), the process mining field has evolved over the past 20 years into a broad and diverse research discipline.

The results of process discovery and consecutive analyses are often directly based on a sample of event data that may not have captured all possible/actual behavior correctly or completely. Therefore, the question whether they also apply to the real, underlying process typically remains unanswered. In order to resolve this, there is a need for unbiased estimators of a discovered model's quality as a representation of the underlying process. The adequacy of the established quality dimensions fitness, precision and generalization is typically only demonstrated using a limited set of special cases, such as flower models or models enumerating one or more traces (Rozinat and van der Aalst 2008; van Dongen et al. 2016). Hence, a critical analysis of these classical dimensions, both on theoretical and empirical grounds, is missing and certainly necessary for process discovery to evolve towards a mature research discipline.

In this paper, we extend the established distinction between exploratory and confirmatory data analysis from traditional statistics to process discovery. As a result,

- we propose a new paradigm to quantify the quality of discovered process models, depending on the type of analysis, and discuss its necessity,

Table 1 Overview of existing quality metrics for fitness (F), precision (P) and generalization (G)

	Metric	Author	Date	Range	Model input type	Included
F	Parsing measure	Weijters et al. (2006)	2006	[0, 1]	Heuristics net	
	Continuous parsing method	Weijters et al. (2006)	2006	[0, 1]	Heuristics net	
	Completeness	Greco et al. (2006)	2006	[0, 1]	Workflow schema	
	Partial fitness – complete	de Medeiros (2006)	2007	$[-\infty, 1]$	Heuristics net	
	Token-based fitness	Rozinat and van der Aalst (2008)	2008	[0, 1]	Petri net	•
	Proper completion	Rozinat and van der Aalst (2008)	2008	[0, 1]	Petri net	
	Negative event recall	vanden Broucke et al. (2014)	2009	[0, 1]	Petri net	•
	Behavioral profile conformance	Weidlich et al. (2011)	2011	[0, 1]	Petri net	
	Alignment-based fitness	van der Aalst et al. (2012)	2012	[0, 1]	Petri net	•
P	Soundness	Greco et al. (2006)	2006	[0, 1]	Workflow schema	
	(Advanced) Behavioral appropriateness	Rozinat and van der Aalst (2008)	2008	[0, 1]	Petri net	
	Behavioral specificity	Goedertier et al. (2009)	2009	[0, 1]	Petri net	
	ETC-precision	Muñoz-Gama and Carmona (2010)	2010	[0, 1]	Petri net	
	Alignment-based precision	van der Aalst et al. (2012)	2012	[0, 1]	Petri net	•
	Negative event precision	vanden Broucke et al. (2014)	2014	[0, 1]	Petri net	•
	One align precision	Adriansyah et al. (2015)	2015	[0, 1]	Petri net	•
	Best align precision	Adriansyah et al. (2015)	2015	[0, 1]	Petri net	•
G	Alignment-based generalization	van der Aalst et al. (2012)	2012	[0, 1]	Petri net	•
	Frequency of use	Buijs (2014)	2014	[0, 1]	Process tree	
	Negative event generalization	vanden Broucke et al. (2014)	2014	[0, 1]	Petri net	•

Based on Janssenswillen et al. (2017)

- we inventorize the state-of-the-art quality metrics and relate them to the proposed perspectives, and
- we empirically analyze the difference between the perspectives and investigate possible biases when using metrics for a different purpose than the one they were designed for.

In the next section we discuss some related work, whereafter the distinction between exploratory and confirmatory analysis is made, both in its traditional context and in a process discovery context. Section 4 takes this distinction further to introduce different sets of measures for quality measurement in process discovery. This section also introduces the problem statement underlying the empirical experiment described in the remainder of the paper. The existing quality metrics are discussed in more detail in Sect. 5. Subsequently, an empirical study has been conducted, of which the methodology is laid out in Sect. 6, the results shown in Sect. 7, and its implications are discussed in Sect. 8. Section 9 concludes the paper.

2 Related Work

The quality of discovered process models is typically characterized by four dimensions: fitness, precision, generalization, and simplicity (van der Aalst 2016). While the first three dimensions all compare the behavior of the event log with the model, simplicity only takes into account the

model. Consequently, simplicity will not be considered in the remainder of this paper.

By far the most studied quality dimension is fitness (de Medeiros 2006; Greco et al. 2006; Rozinat et al. 2007; van der Aalst et al. 2012; vanden Broucke et al. 2014; Weidlich et al. 2011; Weijters et al. 2006). A model with a good fitness allows for the behavior seen in the event log. A good fitness is often regarded as a primary requirement, before considering the other metrics.

Secondly, a model is precise if it does not allow for too much unrecorded behavior. Precision has also received a reasonable amount of attention in literature (Adriansyah et al. 2015; Goedertier et al. 2009; Muñoz-Gama and Carmona 2010; vanden Broucke et al. 2014).

Finally, a model should generalize and not restrict behavior towards the examples seen in the event log. In contrast to fitness and precision, only limited work on generalization is available (van der Aalst et al. 2012; vanden Broucke et al. 2014). Furthermore, the precise definition of the concept is still unclear, as there are multiple interpretations which differ in slight but important ways (van der Aalst 2013; Buijs 2014; vanden Broucke et al. 2014).

Over the last decades, several metrics have been implemented to measure these quality dimensions. For a comprehensive overview of these metrics, we refer to Table 1 and Janssenswillen et al. (2017). The state-of-the-art metrics will be further introduced in Sect. 5.

The dimensions were first introduced in Rozinat et al. (2007) and their adequacy has since received limited critical consideration. In van der Aalst (2013), the focus is on the relation between modeled and recorded behavior. Although the paper emphasizes that *process discovery aims to tell something about the unknown real process*, it states that fitness and precision metrics measure the fit between the model and the event log, while generalization quantifies the quality of these metrics as estimators of fit between system and event log. Unfortunately, the discussion in van der Aalst (2013) is restricted to a theoretical one and is not experimentally validated.

A recent comparative study of process metrics (Janssenswillen et al. 2017) shows that the role of generalization in measuring conformance is extremely ambiguous. The generalization metrics were found to be uncorrelated, with one of them appearing to be related to fitness.

A quite different approach is undertaken in Rogge-Solti et al. (2016). In this study, the authors acknowledge that neither log nor model (be it discovered or designed) provide an accurate description of the underlying process. In order to find a representation of the latter, both log and model are modified by taking into account a certain *trust* in each of them. However, as the approach uses the existing metrics for fitness, precision, and generalization, the accuracy of the result will depend on the quality of these metrics. As the approach is only validated on real-life event logs (where the underlying process is unknown), it is not clear whether the approach succeeds in finding the system.

It is frequently conjectured that the four quality dimensions should not be optimized simultaneously, but that trade-offs exist between the metrics which have to be resolved based on the objective of the analysis (Buijs et al. 2012). However, no guidelines exist on how this trade-off should be solved in a given situation.

In the remainder of this paper, we aim to cast a new light upon these dimensions and metrics by drawing an analogy with the difference between exploratory and confirmatory analysis within traditional statistics.

3 Exploratory and Confirmatory Analysis

3.1 Traditional Data Science

The data science field largely originated from the discipline of statistics during the last decades of the twentieth century (Tukey and Wilk 1966). Within statistics, the emphasis has historically been on confirmatory analysis, relying on the well-known paradigms of testing and estimation (Gelman 2004), to *confirm* or reject a stated hypothesis. However, confirmatory techniques are not designed to find hypotheses. Only when one has a certain clearly formed idea or

hypothesis and data which can be exploited to elucidate that idea, can confirmatory statistics be used to investigate whether or not the idea is justified in the light of the evidence (Erickson and Nosanchuk 1992).

With the arrival of more computational power, and the increase of readily available data, the field of exploratory data analysis (EDA) emerged (Tukey 1977). Exploratory analyses are typically the starting point for a line of research, when no specific statistical hypotheses are specified. It mainly encompasses methods to plot your data and transform it. Even when the question to be answered is perfectly clear, the analysis can benefit from exploratory analysis to test whether underlying assumptions for the confirmatory tests are met and by highlighting and subsequently neutralizing other variables which might have an impact on the question asked.

Exploratory and confirmatory methods are not each other's competitors, but rather go hand in hand. Exploratory analysis will both lead to new ideas to be tested, and perhaps new data to be collected. Moreover, it will form the groundwork for the confirmatory analysis. In confirmatory analysis, it is investigated whether the insights learned from the sample can be applied to the population as a whole. While confirmatory analysis can be seen as the work conducted in a law court to determine guilt based on evidence, exploratory analysis can be seen as the indispensable detective work that has to be performed in advance. Data are explored to find clues, get ideas, and follow up on them in search for new hypotheses (Erickson and Nosanchuk 1992). It is clear that one cannot exist without the other, but they are complimentary and can be used in alternation or parallel.

3.2 Exploring and Confirming Within Process Discovery

Process mining started to emerge at the end of the last century, with pioneering works on the discovery of control-flow from event logs (Agrawal et al. 1998; Cook and Wolf 1995; Datta 1998). Reasons for the emergence of this discipline were the accelerating boost of the data science field and the availability of event-based data, which together have the potential to deliver a highly competitive edge to the process-centric companies of the twentyfirst century.

The concept of a *sample* from statistics finds its equivalent in process mining as the *event log L*. On the other hand, we define a *system S* (Buijs 2014) as the *population* of process behavior. The system thus refers to the underlying process, the way work is done. Just as in traditional statistics, the system and event log are not equal, as the event log is only a sample and can contain noise, i.e., measurement errors and inaccuracies. This is shown conceptually in Fig. 1, originally introduced in Buijs (2014).

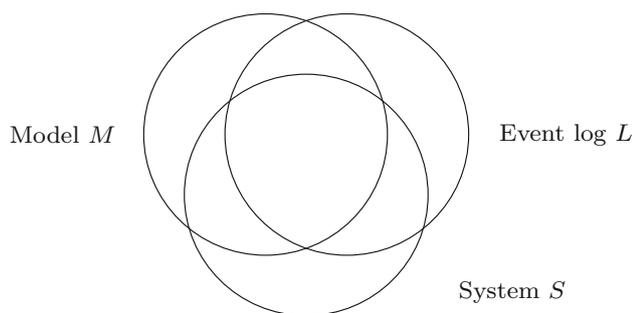


Fig. 1 Venn diagram representing the behavior in the model M , event log L and system S Buijs (2014)

In Fig. 1, the process model M is also represented. This can be either a model designed by the process owners or discovered from event data. But even when the model is learned from the event log, both are typically not equivalent. In order to quantify the quality of a process model to represent a process, different quality dimensions and associated measures implementing these dimensions have been defined. However, Fig. 1 points out that the *quality* of a model can have different interpretations. Given the fact that we can approach the event log as a sample and the system as the population, we can distinguish between an *exploratory* and *confirmatory* approach.

When a confirmatory analysis is intended, it is important that the model used is a correct representation of the system, i.e., $M = S$. For a descriptive, exploratory analysis, this is not the case. In the latter situation, the model used for analysis should have a close fit with the data, i.e., $M = L$.

Just as with traditional exploratory and confirmatory analysis, these two perspectives on the quality of a discovered process model do not need to be in competition with each other. Nevertheless, each of them requires a different conformance checking approach. The next section will thus introduce different quality perspectives for process discovery, each with its own specific metrics.

4 Quality Perspectives for Process Discovery

In this section, the different perspectives towards process quality are introduced formally. In order to do this, some preliminaries are needed.

4.1 Preliminaries

4.1.1 Activity Sequences

Let \mathcal{A} be the activity alphabet. $\mathcal{T} = \mathcal{A}^*$ is the set of all finite sequences over \mathcal{A} , representing the universe of activity sequences. An activity sequence, or trace, $\sigma \in \mathcal{T}$ is a finite

sequence of activities $\langle a_1, \dots, a_n \rangle$. $|\sigma| = n$ refers to the number of activities in a trace. $h(\sigma, k)$ refers to the activity sequence prefix of the first k activities in trace σ . $h(\sigma, 0)$ refers to the empty trace \emptyset .

4.1.2 Event Log

An event log L is a multiset of activity sequences, and can be defined as $L \in \mathbb{B}(\mathcal{T})$, where $\mathbb{B}(\mathcal{T})$ is the set of all multisets of \mathcal{T} . The support of L , denoted as $\text{supp}(L)$, is the set of unique activity sequences in L . Note that $\text{supp}(L) \subseteq \mathcal{T}$. For an activity sequence σ , the frequency of this trace in event log L is defined as $L(\sigma)$. The number of distinct activity sequences in an event log is defined as $|L|$. $\mathbf{L} = \mathbb{B}(\mathcal{T})$ represents the domain of all possible logs.

4.1.3 Model

A model M is a subset of the universe of activity sequences, and can be defined as $M \subseteq \mathcal{T}$. $|M|$ indicates the number of activity sequences part of the model. $\mathbf{M} = \mathbb{P}(\mathcal{T})$ represents the domain of all possible models, where $\mathbb{P}(\mathcal{T})$ is the powerset of \mathcal{T} .

4.1.4 System

A system is defined as a subset of the universe of activity sequences, and can be defined as $S \subseteq \mathcal{T}$. $|S|$ indicates the number of activity sequences part of the system. $\mathbf{S} = \mathbb{P}(\mathcal{T})$ represents the domain of all possible systems.

Using the concepts of log, model and system, we can now formalize different conceptual quality metrics, both for exploratory process discovery and confirmatory process discovery.

4.2 Model-Log Similarity

In the case of exploratory analysis, it is important that there is a close correspondence between the event log and the model. The fit between an event log and a process model is monitored by two ratios (Buijs 2014), *log fitness* and *log precision*. Given event log L , the *log fitness* and *log precision* of a model M can be defined as follows. In these definitions, we assume that the amount of behavior in S , M and $\text{supp}(L)$ is *countable*, which is reflected by a count function $\#(\dots)$.

Definition 1 (*Log fitness*) Log fitness is a function $F^L : \mathbf{M} \times \mathbf{L} \rightarrow [0, 1]$, which quantifies how much of the behavior in the event log is captured by the model. This can be defined conceptually as (Buijs 2014):

$$F^L = F^L(M, L) = \frac{\#(\text{supp}(L) \cap M)}{\#(\text{supp}(L))} \quad (1)$$

Definition 2 (*Log precision*) Log precision is a function $P^L : \mathbf{M} \times \mathbf{L} \rightarrow [0, 1]$, which quantifies how much of the behavior in the model was recorded in the event log. This can be defined conceptually as (Buijs 2014):

$$P^L = P^L(M, L) = \frac{\#(\text{supp}(L) \cap M)}{\#(M)} \quad (2)$$

Only when both log fitness and log precision are equal to 1, then $\text{supp}(L) = M$, i.e., the event log and the model represent exactly the same behavior. These metrics are orthogonal to each other, which makes it possible to construct models which score poorly on one criterion and excellent on the other. Acting as complementary forces, maximizing log fitness and log precision simultaneously maximizes the *fit* between the model and the event log.

4.3 Model-System Similarity

For confirmatory analysis, one would like to reject or *accept* hypotheses such as *Model M_1 is more likely than Model M_2 to be the real underlying system*. In order to do this, it is necessary to estimate how well a model M represents the system S .

When drawing the analogy, it is evident that two similar dimensions are needed to quantify the match between the model and the system. Firstly, there is a need for a metric that ensures the selection of models that contain all possible real behavior. Secondly, a metric that favors the selection of models that only contain real behavior is required. Therefore, given the system S , the *system fitness* and *system precision* of a model M can be defined as:

Definition 3 (*System fitness*) System fitness is a function $F^S : \mathbf{M} \times \mathbf{S} \rightarrow [0, 1]$, which quantifies how much of the behavior in the system is captured by the model. This can be defined conceptually as (Buijs 2014):

$$F^S = F^S(M, S) = \frac{\#(S \cap M)}{\#(S)} \quad (3)$$

Definition 4 (*System precision*) system precision is a function $P^S : \mathbf{M} \times \mathbf{S} \rightarrow [0, 1]$, which quantifies how much of the behavior in the model is part of the system. This can be defined conceptually as (Buijs 2014):

$$P^S = P^S(M, S) = \frac{\#(S \cap M)}{\#(M)} \quad (4)$$

4.4 Problem Statement

In a real-life process mining project, there is an inherent difference between log-measures and system-measures because of sampling errors and observational errors. Given the complexity of business processes, it is unlikely that all the possible behavior and dependencies in a process can be recorded in a reasonable time span. As a result, log precision might be lower than the system precision because the model allows for unrecorded but correct behavior. On the other hand, there can be measurement errors in the data. These can lead to a log fitness which is lower than system fitness, because the model is penalized for not being able to replay behavior which turns out to be incorrect. Furthermore, measurement errors can have an opposite impact on precision, and sampling error can have an opposite impact on fitness. However, system-based metrics cannot be computed since the system is generally unknown in reality. As a result, the question is whether the existing log-based metrics are good estimators of their system-based counterparts. To this end we define

$$\Delta F(L, M, S) = F^L(M, L) - F^S(M, S) \quad (5)$$

ΔF can be computed for each of the existing fitness metrics. For example, to investigate the quality of Token-based Fitness as an estimator of system fitness, we inspect $\Delta F_{tb}(L, M, S) = F_{tb}(M, L) - F_{tb}(M, S)$. By using the Token-based metric itself in the calculation of the system fitness, any metric-dependent effects are ruled out.

The same analysis is conducted for precision, where we define ΔP as

$$\Delta P(L, M, S) = P^L(M, L) - P^S(M, S) \quad (6)$$

Using an empirical analysis, we examine whether the existing quality log-based metrics are indeed unbiased estimators of system-quality. Formally, the next two hypotheses are tested for each existing metric:

$$H_0 : \Delta F = 0 \quad H_1 : \Delta F \neq 0 \quad (7)$$

$$H_0 : \Delta P = 0 \quad H_1 : \Delta P \neq 0 \quad (8)$$

In the next section, we further introduce the existing metrics which are considered in the analysis. The methodology of the empirical examination is detailed in Sect. 6.

5 Existing Quality Metrics

Based on the list of existing metrics in Table 1, nine metrics are considered, as indicated in the last column of Table 1. The selection of this set of metrics is based on the following criteria:

1. They accept a Petri Net as input
2. They return a single value on a $[0, 1]$ scale
3. They can cope with imperfect inputs (unsound discovered models, unfitting logs, etc.)

These criteria should not be interpreted as strict desirable properties of metrics, but rather as practical restrictions needed for a large-scale empirical analysis.

5.1 Fitness

Token-Based Fitness (Rozinat and van der Aalst 2008) (from here on referred to as F_{tb}) is one of the first fitness metrics that was defined. As the name suggests, it is highly dependent on the Petri Net representation of the model under consideration. The metric penalizes both when tokens are missing, i.e., an recorded activity cannot be replayed, and when tokens remain in the model after replay. While the first penalty takes into account whether an activity sequence from the log is part of the model, the latter penalty makes sure that the requirement of proper completion is taken into account. Formally, Token-based Fitness is computed as follows:

$$F_{tb} = \frac{1}{2} \left(1 - \frac{\sum_{\sigma \in \text{supp}(L)} L(\sigma) \cdot m_M(\sigma)}{\sum_{\sigma \in \text{supp}(L)} L(\sigma) \cdot c_M(\sigma)} \right) + \frac{1}{2} \left(1 - \frac{\sum_{\sigma \in \text{supp}(L)} L(\sigma) \cdot r_M(\sigma)}{\sum_{\sigma \in \text{supp}(L)} L(\sigma) \cdot p_M(\sigma)} \right) \quad (9)$$

where $m_M(\sigma)$ refers to the number of missing tokens when replaying trace σ on model M . c , r , and p refer to consumed, remaining and produced tokens, respectively.

Alignment-Based Fitness (van der Aalst et al. 2012) (from here on referred to as F_{ab}) is a fitness metric which differs from Token-based Fitness in that it does not rely on the notion of tokens flowing through a Petri Net. Instead, it *aligns* log and model in terms of activities. This means that for non-fitting traces, i.e., $\{\sigma \mid \sigma \in \text{supp}(L) \wedge \sigma \notin M\}$, the algorithm looks for the execution path in the model which is most alike, as measured by a cost function. The result is an alignment λ between the log trace and the model trace, which by default has a cost of 1 for each insertion and 1 for each deletion.¹ Formally, the total cost of aligning a log and a model is defined as

$$f_{cost} = \sum_{\sigma \in \text{supp}(L)} \delta(\sigma, M) \cdot L(\sigma) \quad (10)$$

where $\delta(\sigma, M)$ is the minimal alignment cost of activity sequence σ with model M . Given this cost function, the Alignment-based Fitness is defined as follows:

$$F_{ab} = 1 - \frac{f_{cost}}{\sum_{\sigma \in \text{supp}(L)} (L(\sigma) \cdot |\sigma| + (L(\sigma) \cdot \min_{\tau \in M} |\tau|))} \quad (11)$$

Note that the denominator of F_{ab} is equal to the maximum possible cost: the number of events in the event log and the number of activities in the shortest path of the model times the number of cases in the event log.

Note that the Alignment-based Fitness is very similar to Token-based Fitness, except for the fact that it counts inserted and deleted activity instances, instead of missing and remaining tokens.

Negative Event Recall (Goedertier et al. 2009) (from here on referred to as F_{ne}), also known as *Behavioral recall*, is different from Token-based and Alignment-based Fitness, in that it uses the notions of precision and recall, known from the field of information retrieval and binary classification. If we define True Positives (TP) as the number of events in the log that can be correctly replayed, and False Negatives (FN) as the number of events in the log for which a transition was forced to fire, Negative Event Recall can be defined as follows:

$$F_{ne} = \frac{TP}{TP + FN} \quad (12)$$

Note that this formula is the same as the well-known formula for recall in binary classification. In this case, the log is regarded as the *true condition* while the model is regarded as the *predicted condition*. The negative event conformance metrics are based on the induction of artificial negative events. However, the negative events only impact the negative event precision and generalization metrics, which will be addressed further on.

Just as Alignment-based and Token-based Fitness, the Negative Event Recall relies only on the log as the single version of the truth. It differs from the other fitness metrics, as it does not penalize improper completion.

5.2 Precision

Alignment-Based Precision (van der Aalst et al. 2012) (from here on referred to as P_{ab}) computes the precision of a model based on the same concept of alignments such as Alignment-based Fitness. It starts from an *aligned* log, in which all the non-fitting traces are replaced with (one of) their optimal alignment(s).² Based on this event log, it considers the activity prefix $h(\sigma, k)$ of each event, and counts which activities are *enabled* in the model after this activity prefix ($en_M(h(\sigma, k))$), and which occurred in the

¹ In practice, these costs can be configured for each activity type individually, to reflect that certain deviations should be penalized more than others.

² Optimal alignments are the alignments for which the cost is minimized.

log after this activity prefix ($en_L(h(\sigma, k))$). It follows that precision is defined as:

$$P_{ab} = \frac{\sum_{\sigma \in \text{supp}(L)} L(\sigma) \sum_{j=0}^{|\sigma|-1} \frac{en_L(h(\sigma, j))}{en_M(h(\sigma, j))}}{\sum_{\sigma \in \text{supp}(L)} |\sigma| \cdot L(\sigma)} \tag{13}$$

The precision measured by this formula will decrease when for one or more activity prefixes more activities are enabled in the model than occurred in the log.

Negative Event Precision (vanden Broucke et al. 2014) (from here on referred to as P_{ne}) is a precision metric which is related to Negative Event Recall, and is also called Behavioral precision. Just like recall, its formula equals the well-known precision formula from the field of binary classification.

$$P_{ne} = \frac{TP}{TP + FP} \tag{14}$$

In this case, False Positives (FP) are events which are allowed by the model but should not be, as their real condition is negative. However, since negative events are not available in process discovery, they have to be induced artificially. The creation of artificial negative events is discussed in Goedertier et al. (2009). During the induction of negative events, a confidence for each negative event is also calculated, which makes it possible to compute a weighted negative event precision.

One-Align Precision (Adriansyah et al. 2015) (from here on referred to as P_{oa}) is a combination of ETC-precision (Muñoz-Gama and Carmona 2010) and alignments (van der Aalst et al. 2012). ETC-precision, or precision based on escaping edges, is a precision metric which constructs an automaton of the behavior in the log. Subsequently, it looks for *escaping edges*, which essentially are events that are allowed by the model in a certain state, but which are never recorded. The precision is then defined as follows,

$$P_{etc} = 1 - \frac{\sum_{\sigma \in \text{supp}(L)} \sum_{j=0}^{|\sigma|-1} |E(h(\sigma, j))|}{\sum_{\sigma \in \text{supp}(L)} \sum_{j=0}^{|\sigma|-1} |A(h(\sigma, j))|} \tag{15}$$

where $E(h(\sigma_i, j))$ refers to the number of escaping edges after activity j of trace σ_i , and $A(h(\sigma_i, j))$ refers to the number of allowed tasks (both recorded activities and escaping edges).

Since the ETC-precision itself requires that the event log has a perfect fitness, it will not be considered further in this paper. However, *One-align Precision* or *Best-align Precision* are used instead, which use an aligned log to compute ETC-precision (Adriansyah et al. 2015).

One-align Precision refers to the application of $P_{etc}(L_a, M)$ where L_a is an aligned log using *one* optimal alignment for each non-fitting trace. Note that more than one optimal alignment can be available for a certain trace.

Best-Align precision (Adriansyah et al. 2015) (from here on referred to as P_{ba}) is similar to One-align Precision, with the only difference that it does not use one alignment but all the optimal alignments for each trace.

5.3 Generalization

Alignment-Based Generalization (van der Aalst et al. 2012) (from here on referred to as G_{ab}) was the first generalization metric to be implemented, and uses trace alignments just like the related fitness and precision metrics. It starts from an aligned log, and for each event calculates the probability that the next time this state is visited, a new path will be recorded. Given n as the number of unique activities enabled in this state, and f as the number of times the state was visited, the probability is defined as

$$p_{new}(n, f) = \begin{cases} \frac{n(n+1)}{f(f-1)}, & \text{if } f - n \geq 2 \\ 1, & \text{otherwise} \end{cases} \tag{16}$$

For example, in a state with 2 unique activities and 2 visits, $p_{new} = 1$, as is also the case with 3 visits. If $f = 4$, $p_{new} = \frac{2 \cdot 3}{4 \cdot 3} = 0.5$. If $f = 5$, $\frac{2 \cdot 3}{5 \cdot 4} = 0.3$. The larger the difference between the number of visits and the number of unique activities, the lower the probability. If the average probability over the log is low, then generalization is assumed to be high. As such,

$$G_{ab} = 1 - \frac{\sum_{\sigma \in \text{supp}(L)} \sum_{j=0}^{|\sigma|-1} p_{new}(en_L(h(\sigma, j)), f(h(\sigma, j)))}{\sum_{\sigma \in \text{supp}(L)} |\sigma| \cdot L(\sigma)} \tag{17}$$

where $en_L(h(\sigma, j))$ is the number of activities are *enabled* in the model after this activity prefix and $f(h(\sigma, j))$ is the frequency with which this state is visited in the log.

Relating this definition to one of the concepts introduced in Sect. 4 is not a trivial task. It tends to favor models in which more activities are possible in a specific state than those which actually occurred in the log. However there is no indication that this additional behavior is real (i.e., belongs to the system, thereby increasing system fitness). Nor is there any upper limit, which means that the flower model will have a perfect generalization according to this metric.

Negative Event Generalization (vanden Broucke et al. 2014) (from here on referred to as G_{ne}), also called Behavioral Generalization, is related to Behavioral recall and precision and relies on the induction of artificial negative events. Negative Event Generalization is defined as

$$G_{ne} = \frac{AG}{AG + DG} \quad (18)$$

where AG refers to the number of *allowed generalizations* and DG refers to the number of *disallowed generalizations*. Generalized events are events which were not recorded but at the same time not considered as negative. In other words, they are supposed to reflect real behavior and thus belong to the system S . Consider system S^* as defined by the induced negative events as an approximation of the real system S . The complete number of generalized events, $AG + DG$ is thus equal to $|S^* \setminus L|$. Generalized events which can be replayed by the model are called *allowed generalizations*, i.e., $AG = |M \cap S^* \setminus L|$. Disallowed generalized events are generalized events which are not allowed by the model, i.e., $DG = |S^* \setminus (L \cup M)|$. This means that G_{ne} can be rewritten as

$$G_{ne} = \frac{|M \cap S^* \setminus L|}{|S^* \setminus L|} \quad (19)$$

which resembles the formula for system fitness, with the only difference that S is replaced by $S^* \setminus L$.

6 Methodology

In order to analyze the quality of the introduced metrics as unbiased estimators of the fit between a discovered model and the underlying system, an experiment is conducted consisting of the following steps:

1. Generate systems
2. Calculate number of paths
3. Simulate logs
4. Discover models
5. Measure log-quality
6. Measure system-quality
7. Statistical analysis

A schematic overview of the methodology is shown in Fig. 2. The different steps are discussed in more detail in the following paragraphs.

6.1 Generate Systems

Firstly, 10 different systems were created. These can be regarded as the real process underlying 10 different business processes. The systems were generated using the methodology in Jouck and Depaire (2016). Process trees were chosen as notation because they can represent all block-structured models. Furthermore, the methodology in Jouck and Depaire (2016) allows to generate process trees with long-term dependencies using unfolded choice trees. Moreover, process trees lend themselves well for this large-

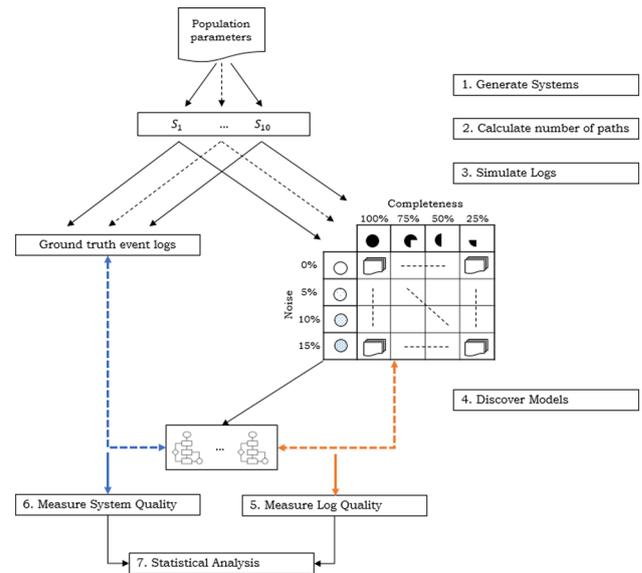


Fig. 2 Schematic overview of methodology

scale experiment, as they are guaranteed to be free of deadlocks.

Ten process trees were generated, each from a different population with another probability distribution for the type of operators (choice, and, loop, etc.), as well as different probabilities for the number of duplicate tasks, silent tasks, long-term dependencies, etc. An overview of the population parameters is shown in Table 2.

The first three parameters define a triangular distribution from which the number of visible activities is randomly drawn. The next five parameters - Π^{\rightarrow} , Π^{\wedge} , Π^{\times} , Π° and Π^{\vee} - define a probability distribution over the different types of process tree operators: sequence, parallel, exclusive choice, loops, and or choice, respectively. The probability that a silent (invisible) activity is included in an exclusive choice, loop, or choice construct is given by Π^{τ} , the probability that an activity is duplicated is defined by Π^{Re} , and Π^{Lt} gives the probability that a long-term dependency is included between two decision points.

The probabilities for sequence, parallel and choice constructs are based on the work in Kunze et al. (2011). In this work, the occurrence of sequence, exclusive choice, and parallelism in a large set of models is analyzed, which (when normalized to 100%), are on average 46%, 35% and 19%. Population 1–6 can be seen as slight variations of the above mentioned probabilities, while populations 7–10 can be seen as more special cases, including duplicate and silent tasks, long-term dependencies, and atypical probability distributions for constructs. The implications of these settings and their limitations are discussed in Sect. 8.

Table 2 Parameters for 10 model populations (MP) to generate systems

Parameters	Population									
	MP_1	MP_2	MP_3	MP_4	MP_5	MP_6	MP_7	MP_8	MP_9	MP_{10}
Minimum of visible act	10	10	10	10	10	10	10	10	10	10
Mode of visible act	15	15	15	15	15	15	15	15	15	15
Maximum of visible act	20	20	20	20	20	20	20	20	20	20
Sequence (Π^{\neg})	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.45
Parallel (Π^{\wedge})	0.30	0.00	0.15	0.15	0.00	0.10	0.10	0.10	0.10	0.00
Choice (Π^{\times})	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.40
Loops (Π°)	0.00	0.30	0.15	0.00	0.15	0.10	0.10	0.10	0.10	0.00
Or (Π^{\vee})	0.00	0.00	0.00	0.15	0.15	0.10	0.10	0.10	0.10	0.15
Silent act (Π^{τ})	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00
Reoccurring act (Π^{Re})	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00
Lt. dependencies (Π^{Lt})	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00

6.2 Calculate Number of Paths

After the generation of the systems, the maximum number of execution paths in each tree is calculated using the algorithm in Janssenswillen et al. (2016). In order to cope with loop operators, the algorithm assumes that a loop cannot be iterated over more than three times, thereby effectively limiting the number of paths which can be generated by a loop in a realistic manner. This limit ensures that each model has a *finite* number of possible execution paths and is inspired by a *fairness assumption*, meaning that a certain task should not be delayed indefinitely. The number of paths in a model is needed to control the completeness of event logs in the next step, i.e., the simulation of event logs.

6.3 Simulate Logs

For each system, different event logs were simulated using the methodology in Jouck and Depaire (2016). Firstly, a ground truth event log was created for each system. This is an event log with zero noise and 100% completeness (indicated by the number of distinct paths calculated in the previous step). This ground truth event log will be used later to calculate the system-quality of models.

Secondly, event logs with varying levels of completeness and noise are generated. The completeness, in terms of number of distinct traces, varies between 25%, 50%, 75% to 100%. The amount of noise ranges from 0% to 5%, 10% and 15%. Noise is defined as low-frequent incorrect behavior (de Medeiros et al. 2007), and the types of noise which are induced are adapted from Maruster (2003).

To assure that the introduction of noise does not decrease the completeness, noise is not directly added to the event log. Instead, a sample of the event log is taken to which noise is added, that is then combined with the

original event log. The size of the sample is derived from the target noise threshold: to obtain an event log with 15% of noise, a sample of size $x\%$ is needed such that $x/(100+x) = 15\%$. Since the original part of the event log still belongs to the modified event log, completeness does not decrease.

However, it is important to observe that this noise threshold should be regarded as an upper bound. A modified trace, i.e., after introducing noise, can still be correct behavior. Currently, the algorithm used for introducing noise does not explicitly test this. Consequently, while introducing noise will not decrease completeness, as a result of the mechanism described above, it can increase the completeness. This happens when the noisy traces are still system behavior and have not yet been seen in the log. As a result, the completeness threshold should be regarded as a lower bound. This means that both the completeness and the noise threshold are defined in a *conservative* way, i.e., the actual level might be not as bad.

Definition 5 (Noise) Given a trace $\sigma = \langle a_1, a_2, \dots, a_{n-1}, a_n \rangle$, then the following types of noise are defined:³

1. Missing head: remove all activities a_i with $i \in [1, \frac{n}{3}]$
2. Missing body: remove all activities a_i with $i \in [\frac{n}{3} + 1, \frac{2n}{3}]$
3. Missing tail: remove all activities a_i with $i \in [\frac{2n}{3} + 1, n]$
4. Swap tasks: interchange two random activities a_i and a_j with $i \neq j$
5. Remove task: remove random activity a_i

³ The types of noise used have been defined based on existing literature (Maruster 2003). However, for future experiments, a more elaborate reasoning for what qualifies as realistic noise is necessary. For example, the swapping of random activities is not really a realistic event. A detailed discussion of what can be regarded as noise is out of the scope of this paper.

These types of noise have been defined based on the fact that they mimic realistic measurement errors or data inconsistency, due to system failures [1–3, 5] or unsynchronized time registrations [4].

For each combination of noise level (4) and completeness level (4), 5 different logs are generated, resulting in a total of $4 \cdot 4 \cdot 5 = 80$ for each system, or 800 logs in total.

6.4 Discover Models

For each of the 800 event logs, three different models are discovered by way of the Heuristics miner (Weijters et al. 2006), the Inductive miner (Leemans et al. 2013), and the ILP miner (van der Werf et al. 2008). The total number of discovered models is thus equal to 2400. ProM 6.5 was used for the discovery of the process models, and each of the miners was used with the default settings.

6.5 Measure Log Quality

After the event logs have been generated and the models are discovered, the quality metrics discussed in Sect. 5 are applied to each discovered process model and the event log it was learned from. Since there are 2400 process models and 9 quality metrics, this results in a total of 21,600 measurements.

6.6 Measure System Quality

Next to the log-quality, also the system quality of process models is measured. This is done by applying each of the fitness and precision metrics with respect to the ground truth event log for each of the systems, as to compute system fitness and system precision of these models. This means that for each model there are actually 3 system fitness measures and 4 system precision measures.

Note that the ground truth event log of the systems is used for several reasons. Firstly, there are no metrics for quantifying a notion of fitness and precision between two process models, which is solved by representing one of them as an equivalent event log. Secondly, the systems are better candidates to be represented by a ground truth event log than the models, as the latter may not be sound. Deadlocks or livelocks might cause problems when simulating the models. Also, the calculated number of paths (see Sect. 6.2) is essential to assure the ground truth event logs are complete. Calculating the number of paths in the models might not be feasible for all discovered models, as the technique in Janssenswillen et al. (2016) requires block-structuredness, which is not guaranteed by ILP-miner and Heuristic miner. Finally, from the viewpoint of comparing log-measures with system-measures, it appears more logical to use the discovered model in the same

appearance (i.e., as a process model) in both measurements.

6.7 Statistical Analysis

The analysis of the results consists of two parts. The first part analyzes the difference between log fitness and log precision on the one hand, and system fitness and system precision on the other hand. The second part analyzes the relationship between generalization metrics and system fitness.

6.7.1 Log Versus System-Perspective

In order to analyze the difference between log fitness and system fitness, and log precision and system precision, we investigate whether the existing fitness and precision measures can be used as an unbiased estimator for system fitness and system precision, respectively. This means that

$$E[\Delta F] = 0 \quad (20)$$

and

$$E[\Delta P] = 0 \quad (21)$$

regardless of the amount of noise or level of completeness of the log. Recall that ΔF and ΔP are defined as follows:

$$\Delta F(L, M, S) = F^L(M, L) - F^S(M, S) \quad (22)$$

$$\Delta P(L, M, S) = P^L(M, L) - P^S(M, S) \quad (23)$$

The distribution and expected values of ΔF and ΔP under different circumstances in terms of noise and completeness are analyzed both visually and using t-tests.

6.7.2 Generalization

Although the concept of generalization, as discussed in Sect. 2, does not directly match the perspectives proposed in Sect. 4, it is to some extent related to system fitness. As a result, next to log fitness metrics, generalization metrics might be a viable candidate as estimators for system fitness. In order to analyze the quality of generalization metrics as unbiased estimators, we compare their value with system fitness. In this analysis, Alignment-based Fitness is chosen as the reference system fitness, as it is considered as the state-of-the-art fitness-metric. Formally, we define

$$\Delta G(L, M, S) = G^L(L, M) - F_{ab}^S(M, S) \quad (24)$$

The distribution of ΔG is analyzed in the same way as those related to fitness and precision, i.e., both graphically and using t-tests.

7 Results

7.1 Log Versus System-Perspective

7.1.1 Fitness

Figure 3 shows that the influence of completeness and noise on the distribution of ΔF is quite different. Note that in this and subsequent figures, there is a data point for each combination of simulated event log, discovered model, and quality metric used. In Fig. 3a it can be seen that, if the completeness of the log decreases, log fitness measures remain unbiased estimators of system fitness, but their precision as estimator decreases.

On the other hand, when the amount of noise in the event log increases - keeping completeness constant - both the variance of ΔF increases and its expected value decreases. In the presence of noise, log fitness metrics are thus biased estimators of system fitness; they underestimate real system fitness.

Table 3 shows the extent of the biases in more detail for each of the metrics. T-tests were conducted to see whether the mean ΔF was equal to zero or not, under the various circumstances. The annotated *'s indicate whether ΔF is significantly different from zero in a certain situation. In order to correct for multiple testing, the Bonferroni correction was applied. It can be recorded that the impact of incompleteness (in the absence of noise) is limited, with only a few statistically significant differences. However, when the logs contain noise, there are statistically significant underestimations of system fitness.

7.1.2 Precision

Figure 4a shows that when event logs are incomplete, precision measures are increasingly underestimating system precision, while Fig. 4b shows that they overestimate system precision in case of noisy logs.

The mean ΔP for different levels of noise and completeness is shown in Table 4. In this case, both noise and completeness have a statistically significant impact on ΔP .

In general, it can be stated that incompleteness of the event log always leads to an underestimation of system precision, while noise results in an overestimation. However, making assumptions about the completeness and the amount of noise of a given event log is a non-trivial task. As a result, quantifying the bias in a particular case would not be straightforward.

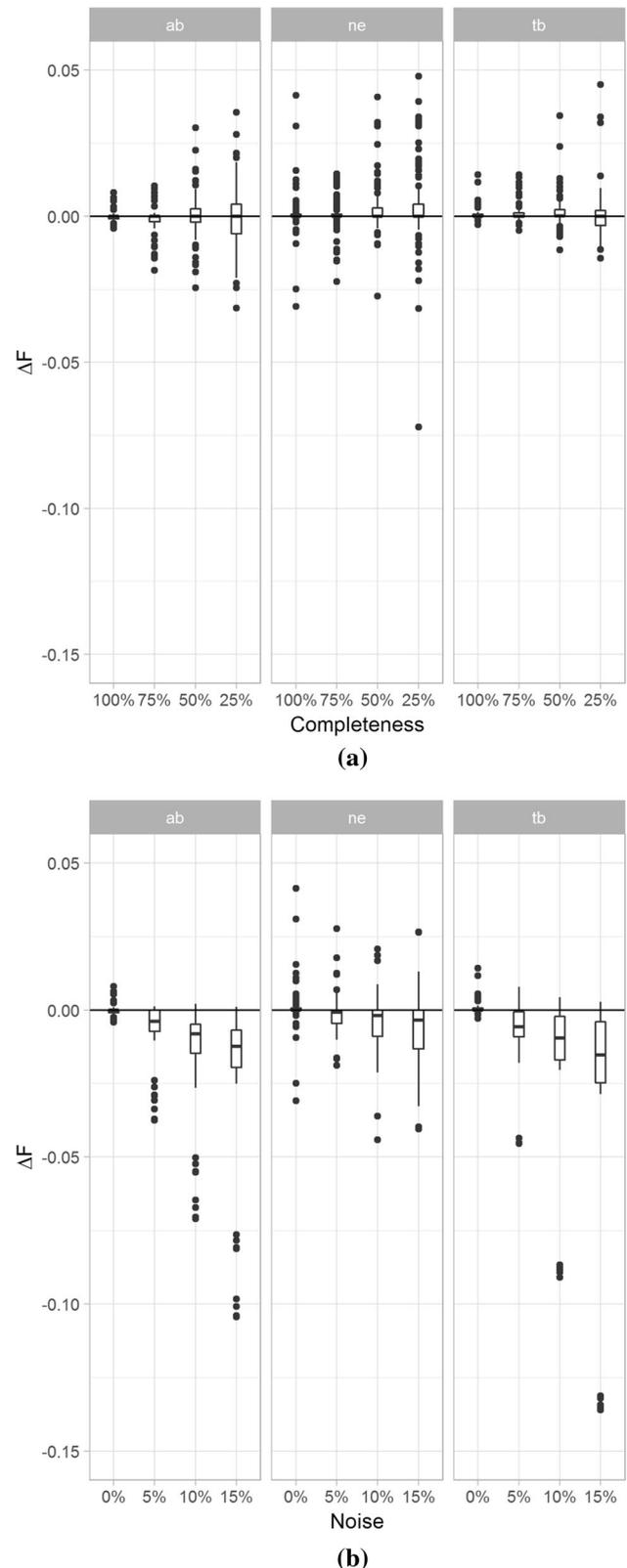


Fig. 3 Impact of completeness and noise on ΔF . **a** Distribution of ΔF for different levels of completeness, while noise is constant at 0%. **b** Distribution of ΔF for different levels of noise, while completeness is constant at 100%

Table 3 Mean ΔF for fitness metrics under differing noise and completeness levels

Metric	Completeness (%)	Noise			
		0%	5%	10%	15%
Alignment-based Fitness	100	− 0.0002	− 0.0071***	− 0.0144***	− 0.0212***
	75	− 0.0013	− 0.0081***	− 0.0158***	− 0.0217***
	50	0.0002	− 0.0066***	− 0.013***	− 0.0209***
	25	0.0011	− 0.0051*	− 0.0115***	− 0.0181***
Negative Event Recall	100	0.0011**	− 0.0017***	− 0.0047***	− 0.0069***
	75	0.0003	− 0.0017***	− 0.0049***	− 0.0076***
	50	0.0024***	− 0.002***	− 0.0043***	− 0.008***
	25	0.0033**	0.0011	− 0.0034***	− 0.0057***
Token-based Fitness	100	0.0007	− 0.0069***	− 0.0155***	− 0.023***
	75	0.0011	− 0.0049***	− 0.0106***	− 0.0195***
	50	0.0016	− 0.0037***	− 0.011***	− 0.017***
	25	0.0024	− 0.0014**	− 0.006***	− 0.0082***

* $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

Based on Wilcoxon signed rank test with Bonferroni correction

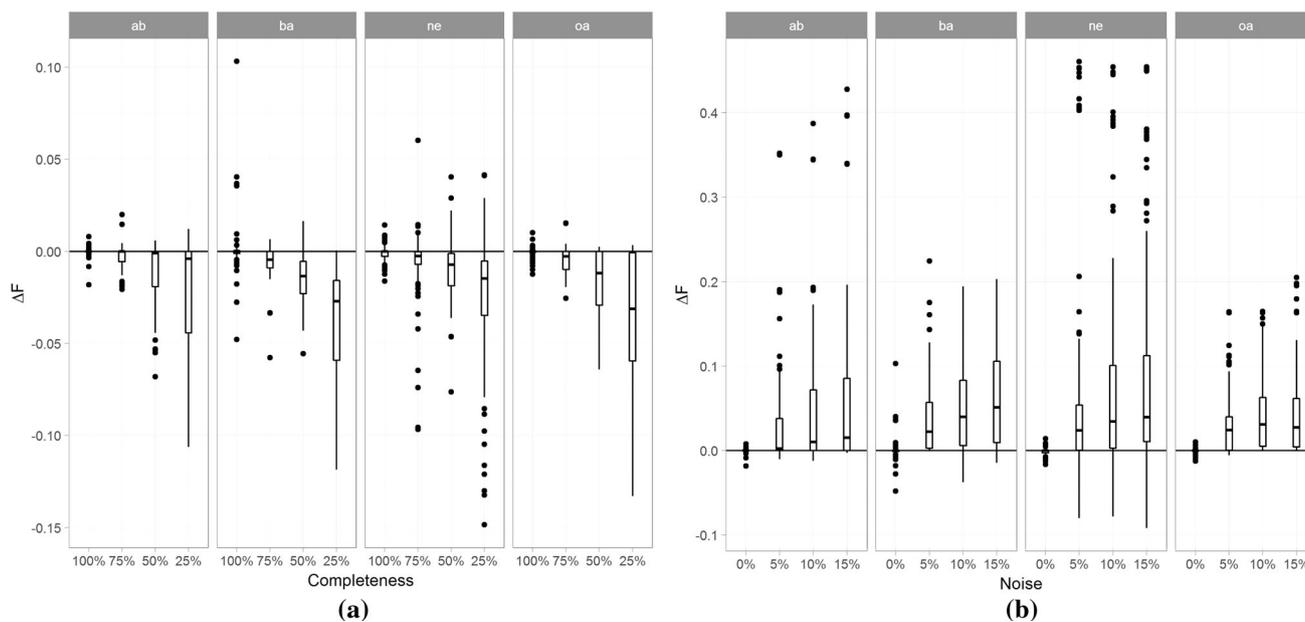


Fig. 4 Impact of completeness and noise on ΔP . **a** Distribution of ΔP for different levels of completeness, while noise is constant at 0%. **b** Distribution of ΔP for different levels of noise, while completeness is constant at 100%

7.2 Generalization

Figure 5 shows the impact of both incompleteness (Fig. 5a) and noise (Fig. 5b) on ΔG . It can be seen that there is a clear distinction between the Alignment-based Generalization and Negative Event Generalization. Although ΔG is more or less stable for both metrics when the completeness of event logs decreases, this is not the case when the amount of noise increases.

Moreover, the impact of noise does not seem to be linear. For Alignment-based Generalization there is a sudden increase in ΔG when the amount of noise is increased from 0% to 5%. As a result, this generalization metric overestimates system fitness. However, when noise increases further than 5%, there is no increase in the overestimation. On the other hand, the pattern for Negative Event Generalization is more erratic, with a strange underestimation for logs with 10% noise, while the bias remains limited at other levels of noise.

Table 4 Mean ΔP for precision metrics under differing noise and completeness levels

Metric	Completeness (%)	Noise			
		0%	5%	10%	15%
Alignment-based precision	100	– 0.0002	0.0415***	0.0453***	0.0597***
	75	– 0.0032***	0.0339***	0.043***	0.049***
	50	– 0.0101***	0.0268	0.0379***	0.0384***
	25	– 0.0225***	0.0018*	0.0093	0.0122
Best-align precision	100	0.0013	0.0412***	0.0538***	0.0636***
	75	– 0.0066***	0.0201***	0.0161***	0.0308***
	50	– 0.015***	0.0085	0.0118	0.0104
	25	– 0.0394***	– 0.015	– 0.0063	– 0.0111
Negative event precision	100	– 0.0012***	0.0595***	0.0728***	0.0837***
	75	– 0.0055***	0.0265**	0.0425***	0.053***
	50	– 0.0101***	0.0157	0.0185	0.0246
	25	– 0.0254***	– 0.0073	– 0.0088	– 0.0047
One-align precision	100	– 0.0004	0.0334***	0.042***	0.0467***
	75	– 0.0049***	0.0174***	0.0262***	0.0315***
	50	– 0.0156***	0.0069	0.012**	0.0152**
	25	– 0.0381***	– 0.0124***	– 0.0064	– 0.0013

* $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$ Based on Wilcoxon signed rank test with Bonferroni correction

The mean values of ΔG in Table 5 show that for both metrics, ΔG is statistically different from zero in nearly all situations where noise or incompleteness is the case. This indicates that Negative Event Generalization consistently underestimates system fitness, even in the absence of noise and for complete event logs.

8 Discussion

When assessing the quality of a process model, often the implicit goal is to find out whether it reflects the underlying, unknown process, on the basis of the sample of event data that has been collected. However, the ability of current metrics to assess the similarity between a process model and the underlying system has never been explicitly tested. As a result, one should be careful when interpreting the obtained measures.

The empirical analysis described in this paper shows that the fitness and precision measures are indeed biased estimators of system fitness and system precision in realistic circumstances, i.e., in the presence of noise and incomplete event data.

Noise leads to an overestimation of system precision and underestimation of system fitness, while incompleteness has the opposite effect. While the direction of the biases are intuitive, the empirical study has shown how severe they are in terms of the level of noise and incompleteness used. Nonetheless, estimating what the amount of noise or the

level of log completeness is in a specific practical context is a difficult task.

It can thus be concluded that, given the metrics which are available today, we are not able to confidently quantify which model is the best representation of the underlying process under consideration, which is definitely an obstacle to evolving towards confirmatory process discovery. It is therefore important not to derive too many conclusions when using fitness and precision metrics, as they only assess the log-perspective.

Nonetheless, information on the direction of the biases, i.e., under- vs overestimation, provides some guidance to practitioners on how to use these obtained quality measures. In case of underestimation, the obtained values can be seen as lower bounds, or conservative measures, while in case of overestimation they should be regarded as being optimistic. A key assumption here is that the practitioner has a good understanding of the noise and completeness of the data used.

The experiment described in this paper has some limitations. Firstly, although the empirical analysis was performed using a set of systems generated with various parameter settings, the instances are too limited to compare the impact of individual parameters on the measurement biases. Further research would be needed to see whether the biases can be linked to characteristics in the process, and thus be analyzed in more detail. Moreover, while the results can be generalized to the populations described in Table 2, additional research is needed to determine

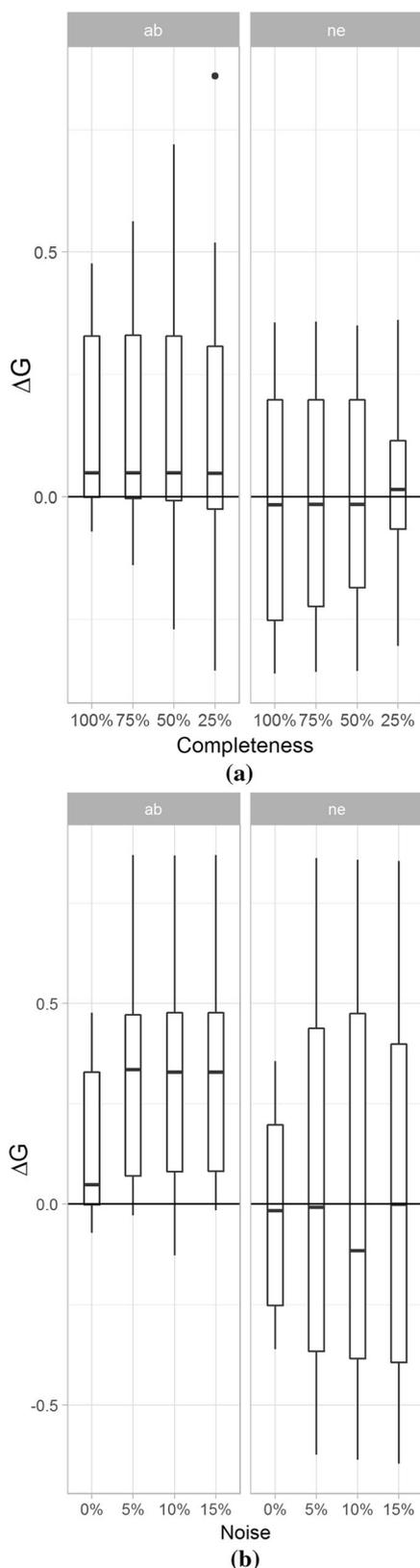


Fig. 5 Impact of completeness and noise on ΔG . **a** Distribution of ΔG for different levels of completeness, while noise is constant at 0%. **b** Distribution of ΔG for different levels of noise, while completeness is constant at 100%

whether these parameters adequately represent realistic process models.

Secondly, since the algorithm for noise induction does not strictly ensure that the resulting traces are incorrect, the noise threshold is an upper bound and the completeness threshold is a lower bound. While this creates difficulties in interpreting the results of the experiment, it is less relevant from a practitioner's point of view, in which the amount of noise and completeness is unknown in any case.

Thirdly, only three discovery algorithms were used in the experiment, each with default settings. While the aim of the experiment was not to compare different algorithms, further research is needed to verify whether the biases can be generalized to other sets of models.

Future research is needed in order to resolve these issues. We believe that additional insights from fields such as statistics and machine learning can facilitate the finding of solutions. Traditional statistical inference could provide answers if event logs are regarded as sets of traces with individual quality measures over which a standard deviation can be computed. Moreover, a promising track for further research would be to compare a set of possible models using Bayesian inference, in order to estimate the likelihood that they represent the underlying system, given the data.

9 Conclusion

Since the emergence of the process mining field, the focus has been largely on exploratory and descriptive data analysis. In other words, the main emphasis was on the sample of event data under consideration, while few efforts have been made to statistically confirm findings. For process discovery to mature as a research field and in order to increase the adoption of process discovery techniques in industry, the latter step is however essential.

In this paper, we connect the process discovery context with the traditional concepts and exploratory and confirmatory analysis in statistics and data science. In particular, when checking the quality of discovered process models, it is important to be aware whether the conclusions of process discovery techniques only apply to the sample of the event data, or conversely apply to the broader context of the process itself. It is shown that new quality dimensions are needed in order to make these kinds of assertions about the system.

An empirical analysis showed that current fitness and precision metrics, which are targeted towards log and model, are biased estimators of the resemblance between model and the underlying system. As a result, although they are sufficient for measuring the quality of a model as a representation of the log, they should not be used when the

Table 5 Mean ΔG under differing noise and completeness levels

Metric	Completeness (%)	Noise			
		0%	5%	10%	15%
Alignment-based generalization	100	− 0.0001**	0.0101***	0.0099***	0.0175***
	75	− 0.0052***	0.0053***	0.0066***	0.0077
	50	− 0.0141***	− 0.0048***	0.0046***	0.0038***
	25	− 0.0291***	− 0.0298***	− 0.0275***	− 0.0278***
Negative event generalization	100	− 0.0054	− 0.244***	− 0.2487***	− 0.2529***
	75	− 0.0075	− 0.2323***	− 0.2527***	− 0.2574***
	50	− 0.0073***	− 0.194***	− 0.2241***	− 0.2431***
	25	− 0.0126**	− 0.1466***	− 0.1807***	− 0.2***

* $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$

Based on Wilcoxon signed rank test with Bonferroni correction

goal is to make statements about the real process. Furthermore, the generalization dimension has been identified as a vaguely defined concept which is unable to properly grasp the relation between model and system. The implemented generalization metrics are moreover unfit for estimating system fitness or system precision.

Acknowledgements The computational resources and services used in this work for both process discovery and process conformance tasks were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government.

References

- Adriansyah A, Munoz-Gama J, Carmona J, van Dongen BF, van der Aalst WM (2015) Measuring precision of modeled behavior. *Inf Syst e-Bus Manag* 13(1):37–67
- Agrawal R, Gunopulos D, Leymann F (1998) Mining process models from workflow logs. In: Schek HJ, Saltor F, Ramos I, Alonso G (eds) *Adv Database Technol - EDBT '98*, vol 1377. Springer, Berlin, pp 467–483
- Buijs JCAM (2014) Flexible evolutionary algorithms for mining structured process models. Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven
- Buijs JCAM, van Dongen BF, van der Aalst WMP (2012) On the role of fitness, precision, generalization and simplicity in process discovery. In: *On the move to meaningful internet systems: OTM 2012*, Springer, Berlin, pp 305–322
- Cook JE, Wolf AL (1995) Automating process discovery through event-data analysis. In: *17th international conference on software engineering*, 1995. ICSE 1995, IEEE, pp 73–73
- Datta A (1998) Automating the discovery of as-is business process models: probabilistic and algorithmic approaches. *Inf Syst Res* 9(3):275–301
- Erickson B, Nosanchuk T (1992) *Understanding data*. McGraw-Hill Education, New York
- Gelman A (2004) Exploratory data analysis for complex models. *J Comput Gr Stat* 13(4):755–779
- Goedertier S, Martens D, Vanthienen J, Baesens B (2009) Robust process discovery with artificial negative events. *J Mach Learn Res* 10:1305–1340
- Greco G, Guzzo A, Ponieri L, Sacca D (2006) Discovering expressive process models by clustering log traces. *IEEE Trans Knowl Data Eng* 18(8):1010–1027
- Janssenswillen G, Depaire B, Jouck T (2016) Calculating the number of unique paths in a block-structured process model. In: *Proceedings of the international workshop on algorithms and theories for the analysis of event data 2016*
- Janssenswillen G, Donders N, Jouck T, Depaire B (2017) A comparative study of existing quality measures for process discovery. *Inf Syst* 71:1–15
- Jouck T, Depaire B (Mar 2016) Generating artificial data for empirical analysis of process discovery algorithms: a process tree and log generator. Technical report, Universiteit Hasselt, Hasselt
- Kunze M, Luebbe A, Weidlich M, Weske M (2011) Towards understanding process modeling—the case of the BPM academic initiative. In: *International workshop on business process modeling notation*, Springer, Berlin, pp 44–58
- Leemans SJJ, Fahland D, van der Aalst WMP (2013) Discovering block-structured process models from event logs—a constructive approach. *Appl Theory Petri Nets Concurr*. Springer, Berlin, pp 311–329
- Maruster L (2003) A machine learning approach to understand business processes. Technische Universiteit Eindhoven
- de Medeiros AKA, Weijters AJ, van der Aalst WMP (2007) Genetic process mining: an experimental evaluation. *Data Min Knowl Discov* 14(2):245–304
- de Medeiros AKA (2006) Genetic process mining. Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven
- Munoz-Gama J, Carmona J (2010) A fresh look at precision in process conformance. In: *Business process management*, vol 6336, Springer, Hoboken, pp 211–226
- Rogge-Solti A, Senderovich A, Weidlich M, Mendling J, Gal A (2016) In log and model we trust? In: *EMISA*, pp 91–94
- Rozinat A, De Medeiros AA, Günther CW, Weijters A, Van der Aalst WM (2007) Towards an evaluation framework for process mining algorithms, vol 123
- Rozinat A, van der Aalst WMP (2008) Conformance checking of processes based on monitoring real behavior. *Inf Syst* 33(1):64–95
- Tukey JW (1977) *Exploratory data analysis*, vol 2. Addison-Wesley, Reading, MA
- Tukey JW, Wilk MB (1966) Data analysis and statistics: an expository overview. In: *Proceedings of the November 7-10*,

- 1966, fall joint computer conference, ACM, New York, pp 695–709
- van der Aalst WMP, Adriansyah A, van Dongen B (2012) Replaying history on process models for conformance checking and performance analysis. *Wiley Interdiscip Rev Data Min Knowl Discov* 2(2):182–192
- van der Aalst WMP (2013) Mediating between modeled and observed behavior: the quest for the “right” process. In: *IEEE international conference on research challenges in information science (RCIS 2013)*, pp 31–43
- van der Aalst WMP (2016) *Process mining: data science in action*. Springer, Berlin
- van der Werf JME, van Dongen BF, Hurkens CA, Serebrenik A (2008) Process discovery using integer linear programming. In: *International conference on applications and theory of petri nets*. Springer, Berlin, pp 368–387
- van Dongen BF, Carmona J, Chatain T (2016) A unified approach for measuring precision and generalization based on anti-alignments. In: *International conference on business process management*. Springer, Cham
- vandenBroucke SKLM, DeWeerd J, Vanthienen Jan B, Baesens B (2014) Determining process model precision and generalization with weighted artificial negative events. *IEEE Trans Knowl Data Eng* 26(8):1877–1889
- Weidlich M, Polyvyanyy A, Desai N, Mendling J, Weske M (2011) Process compliance analysis based on behavioural profiles. *Inf Syst* 36(7):1009–1025
- Weijters AJMM, van Der Aalst WMP, De Medeiros AKA (2006) *Process mining with the heuristics miner-algorithm*. Technische Universiteit Eindhoven, Tech. Rep. WP vol 166, pp 1–34