Kunkler and Rinderle-Ma *Process Science* (2026) 3:3
https://doi.org/10.1007/s44311-025-00036-2

Process Science

RESEARCH Open Access

Probabilistic learning of processing and waiting times: a scalable approach for process event log data

Michel Kunkler[1*] and Stefanie Rinderle-Ma[1]

*Correspondence:
Michel Kunkler
michel.kunkler@tum.de
[1]Technical University of Munich, TUM School of Computation, Information and Technology, Boltzmannstraße 3, 85748 Garching bei München, Germany

Abstract

Business process analysis and improvement approaches typically require accurate models of activity processing or preceding waiting times. Data-driven methods derive these from historical event log data, but typically fit static probability distributions that overlook contextual information or train regression models to produce point estimates, disregarding potential uncertainties. To address these limitations, we present the *Pro3Log* approach that enables Probabilistic learning of Processing and waiting times of activities based on Process event logs. Probabilistic learning is a machine learning approach that enables the learning of dynamic probability distributions, which can model uncertainties while incorporating relevant contextual information. Our *Pro3Log* approach encodes the history of a process case to a fixed-sized input vector and uses DR-BART, a tree-based ensemble model, for deriving dynamic probability distributions. To handle large event logs, *Pro3Log* uses a Mixture of Experts framework where several DR-BART models are trained on a predefined number of subsets of an event log. The subsets are obtained by splitting the event log into clusters using the agglomerative Information Bottleneck algorithm, aiming for homogeneity in processing and waiting times within each cluster. This approach reduces data complexity, allowing for more effective training on each cluster. We apply *Pro3Log* in a business process simulation and demonstrate that it can be used to improve the accuracy of simulation models. These results demonstrate *Pro3Log*'s potential to enhance diverse BPM techniques, including predictive process monitoring or resource allocation approaches.

Keywords Probabilistic learning, Business process simulation, Business process management, Process mining, Waiting times, Processing times

Introduction

Time-related key performance indicators (KPIs) have been acknowledged as one of the most prominent measures of a business process performance (van der Aalst et al. 2016; Van Looy and Shafagatova 2016). Hence, in the field of business process management and process mining, many approaches are concerned with the analysis and optimization of time-related KPIs. One prominent KPI is the *average process cycle time*, i.e., the

average time from the beginning of a process until its completion. This KPI has been analyzed via numerous methods ranging from simple flow analysis (Dumas et al. 2018) to more complex business process simulation approaches (van der Aalst 2015) and has been optimized via, e.g., providing improved resource allocation policies (Pufahl et al. 2025), or recommending process interventions (Weinzierl et al. 2024).

Many of these analyses or optimization methods follow a white-box paradigm where information about a KPI is obtained by inferring and aggregating information about individual process components. E.g., a sample of the process cycle time can be generated by first sampling a sequence of activities until completion, then sampling waiting and processing times for each activity, and finally aggregating these durations (Chapela-Campa et al. 2025). Such a bottom-up analysis demands precise models for each individual process component. This can be challenging because it requires a precise modeling of all types of uncertainties inherent in business processes. Some uncertainties may be intentionally incorporated into the process to enhance its flexibility, for example, by intentionally implementing parallel or alternative branches in an imperative process model (Schonenberg et al. 2008). Other uncertainties caused by external factors, such as those from suppliers, customers, or environmental contexts (e.g., weather or public holidays) (Rosemann et al. 2008), may be undesired and uncontrollable by the organization executing the process. A similar distinction is made in the process quality improvement methodology Six Sigma (Pyzdek and Keller 2024), where it is distinguished between two causes of process variation, namely between *special causes*, which can be controlled or eliminated by a process operator, and *common causes*, which are inherent in the process and generally are not controllable by the operator.

Advanced business process simulation methods often model uncertainties as probability distributions and determine a process outcome by, for example, sampling the branch at a decision point from a categorical probability distribution, or sampling case arrival times or processing and waiting times from continuous probability distributions (Camargo et al. 2020). When such models are used for analyzing or optimizing time-related KPIs, such as the *average process cycle time*, an accurate representation of processing and waiting times is crucial, as the KPI metrics are constituted by a set of these durations (Ali et al. 2025).

While probability distributions are suitable for modeling knowledge about uncertainties (Hüllermeier and Waegeman 2021), the shape of these distributions may be context-dependent, e.g., depending on the current state of the process. To illustrate this, Fig. 1 shows an exemplary process from the manufacturing domain, consisting of three activities.

The first activity *waits for all parts to arrive*, which are delivered by a parcel delivery service that arrives every morning at around 9 a.m. The resulting activity completion times can be described by a multi-peaked probability distribution, as shown below the activity. Assume further that the *quality control* activity can be executed faster every time it is executed again, e.g., because fewer checks have to be done in consecutive executions. Then, a fitting probability distribution would not only depend on the name of the activity but also on its previous number of executions in the running process (see distribution below the activity).

Current data-driven business process simulation approaches, i.e., approaches that learn the parameters for a business process simulation from historical data, often

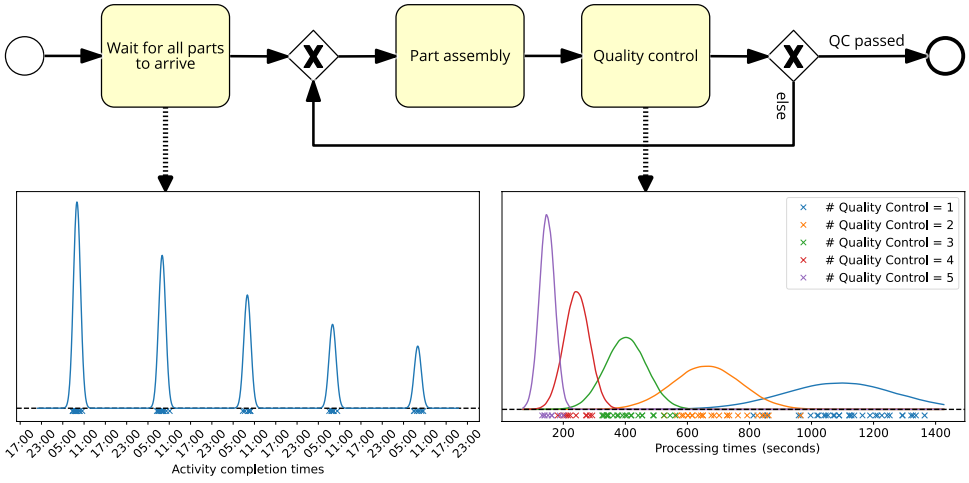

Fig. 1 Historical samples of the processing times of two activities and their underlying probability density functions

overlook the fact that probability distributions for processing and waiting times may be context-dependent. While some approaches have acknowledged the importance of contextual information, such as previous activities, for processing and waiting times, they have overlooked uncertainties by relying solely on a point estimation regression model to obtain a duration. For example, Camargo et al. (2022) use an LSTM for a point estimation of processing times. While such approaches take contextual information into account, they do not provide information about uncertainty, as only a single (point estimation) duration is returned. Other approaches ignore available contextual data and fit static processing and waiting time probability distributions for each activity (see e.g., Rozinat et al. (2009); López-Pintado et al. (2024)). Moreover, in these approaches, often only simple parametric, e.g., single-peaked, probability distributions are fitted to the data, which cannot adequately model the processing times for the *wait for all parts to arrive* activity in our example (Fig. 1). Overall, uncertainties in processing and waiting times should ideally be modeled as context-dependent and non-parametric probability distributions, meaning they should not be constrained to any specific parametric family.

Hence, in this work, we present the *Pro3Log* approach that enables Probabilistic learning of Processing and waiting times of activities based on Process event logs. From a research method point of view, we follow the Algorithmic Engineering methodology as elaborated in Mendling et al. (2025). The *real-world problem* tackled by *Pro3Log* is motivated by the observation that processing and waiting times of activities might depend on their context, e.g., on "multitasking, batching, fatigue effects, and inter-process resource sharing" (Camargo et al. 2022) and additional uncertainty (Weytjens and Weerdt 2022). The *envisioned solution* learns dynamic probability distributions which can, e.g., be applied in simulation models, translated into the *algorithmic task* of taking an event log as input and learning the simulation models with probabilistic distributions of activities. The *algorithmic design* requires an encoding of event logs and a model to learn dynamic probability distributions. The design follows a *deductive approach*, i.e., it employs DR-BART for training dynamic probability distributions for activity processing and waiting times. This primary *design decision* builds on the observation that learning dynamic probability distributions has been addressed in statistics and machine learning as distribution(al) regression (Klein 2024; Kneib et al. 2023) and probabilistic learning

(Klein 2024). A recently proposed probabilistic learning model is Density Regression - Bayesian Additive Regression Trees (DR-BART) (Orlandi et al. 2021). DR-BART is a non-parametric tree-based ensemble model. For given feature data, DR-BART yields a Gaussian Mixture Model (GMM), which can approximate any smooth probability density function to a desired degree of accuracy. Furthermore, due to its tree-based structure, DR-BART can "capture complex, nonlinear relationships and interactions" (Orlandi et al. 2021). This enables DR-BART to learn multi-peaked and context-dependent probability distributions.

As the second *design decision*, we use feature engineering, including inter-case features, and prefix encoding techniques to encode contextual information and the current state of a process instance to a fixed-sized length as required by DR-BART. The observed scalability issues of DR-BART when event logs are large lead to the third *design decision* to present a Mixture of Experts (MoE) model that uses the agglomerative Information Bottleneck (IB) algorithm to partition the input space into a predefined number of clusters and train individual DR-BART models (experts) on the data of each cluster. *Pro3Log* is prototypically implemented, along with three baseline approaches (PIX, Quantile Regression, and Gaussian LSTM) for comparison on one artificial and four real-world event logs. With the design and evaluation choices, we aim to address the design, implementation, and external validity of *Pro3Log*.

The results demonstrate that integrating *Pro3Log* into a business process simulation model enhances the accuracy of simulated process cycle times. This implies that *Pro3Log* might be valuable in other fields that require precise representations of activity processing or waiting times, such as white-box predictive process monitoring, or resource allocation and scheduling approaches.

This work presents an extension to our earlier work (Kunkler and Rinderle-Ma 2025) in which we introduced the basic *Pro3Log* approach for using DR-BART on event log data. The current article extends our previous work in the direction of scalability and practical feasibility. In particular, we introduce an advanced version of *Pro3Log*, which is based on a MoE model utilizing the agglomerative IB algorithm to achieve scalability. Additionally, we provide a numerically stable implementation of DR-BART and present an extended evaluation on two additional data sets. Lastly, we provide an extended evaluation on two additional datasets. For the extended evaluation, we furthermore incorporate inter-instance features and evaluate the numerically stable basic *Pro3Log* implementation, *Pro3Log*'s MoE variant, as well as two additional baseline approaches.

This work is structured as follows. In Section "Preliminaries", we introduce uncertainties and present DR-BART and the agglomerative IB algorithm. We describe how *Pro3Log* encodes event log data in Section "Event log encoding for DR-BART" and how it is trained in Section "Training and inferring from DR-BART models". We describe how *Pro3Log* can be applied in business process simulation in Section "Evaluation" and present evaluation results in Section "Results". In Section "Related work", we present related works on how processing and waiting times of business processes have been modeled. Lastly, we conclude this paper in Section "Discussion & conclusion".

Preliminaries

In this section, we present the fundamental concepts on which *Pro3Log* is based, namely probabilistic learning, DR-BART, Mutual Information (MI), and the agglomerative IB algorithm.

Uncertainties and probabilistic learning

In machine learning, a distinction has been made between two types of uncertainties, i.e., aleatoric and epistemic uncertainties (Hüllermeier and Waegeman 2021). Aleatoric uncertainties are considered irreducible as they stem from inherently random effects. In contrast, epistemic uncertainties are referred to as "uncertainty due to a lack of knowledge about the perfect predictor" (Hüllermeier and Waegeman 2021) and hence are considered reducible uncertainties. Epistemic uncertainties can be further divided into approximation and model uncertainties. Approximation uncertainties refer to uncertainties due to a lack of data for selecting appropriate parameters for a predictor model. In general, approximation uncertainties can be reduced by obtaining more training samples. Model uncertainties refer to uncertainties due to a model's insufficient approximation capabilities. Models with high capacity allow for more flexibility, which can lead to the disappearance of model uncertainties (Hüllermeier and Waegeman 2021). However, approximation uncertainty can be challenging when training models with a high capacity. As models with little capacity often make stronger model assumptions, i.e., stronger assumptions about the underlying data, they may require less data to fit the model.

There has been an ongoing debate about how epistemic uncertainties, i.e., a lack of knowledge, should be modeled (Hüllermeier and Waegeman 2021), and various representations have been proposed (c.f. Destercke et al. (2008)), while aleatoric uncertainties are effectively captured as probability distributions (Hüllermeier and Waegeman 2021).

Learning probability distributions has been addressed in statistics and machine learning as distribution(al) regression (Klein 2024; Kneib et al. 2023), or probabilistic learning (Klein 2024). Klein (2024) distinguishes the term probabilistic learning from distributional regression by its ability to learn higher-order dependencies inherent in the data by employing machine learning techniques. The goal of training probabilistic models is typically to minimize a specific loss function, often based on proper scoring rules (Hüllermeier and Waegeman 2021).

DR-BART

DR-BART is a recently proposed non-parametric Bayesian model (Orlandi et al. 2021) that estimates the full conditional density $p(y|x)$ for a response y given the covariates (features) x. Specifically, the probability distribution returned by DR-BART is a (univariate) GMM, which means that DR-BART is able to approximate any smooth probability density function to a desired degree.

The standard DR-BART model combines two tree-based ensemble models, where the first tree-based ensemble model is used to obtain a mean parameter for a normal distribution and the second tree-based ensemble model is used to obtain a log-variance parameter. In particular, Orlandi et al. (2021) use two sum-of-trees ensemble models, where the mean or log-variance parameter, respectively, are obtained by taking the sum of each tree's leaf value that is identified for the given input data. In the implementation

from Orlandi et al. (2021), the default number of mean trees is set to $m_{\text{mean}} = 200$, and the number of variance trees to $m_{\text{var}} = 100$.[1]

To capture complex, multi-model probability distributions, DR-BART uses a GMM framework. It introduces a latent variable u that indexes the Gaussian components. By marginalizing over u, DR-BART integrates the contribution of multiple Gaussian components to compute the conditional density.

DR-BART models are trained via Bayesian inference. It regularizes the tree structure by requiring a minimum amount of observations for each leaf node and putting priors over the trees' shapes. In particular, it uses two parameters (α, β) to regularize the tree structure. The default default values used in Orlandi et al. (2021) are $(\alpha = 0.95, \beta = 2)$, which aim at rewarding 'bushy' trees (see Chipman et al. (1998)). For approximating the posterior probability distribution, DR-BART uses Markov Chain Monte Carlo (MCMC) with two steps for each iteration: First, DR-BART uses Metropolis-Hastings, where in each Metropolis-Hastings iteration, a change to the structure of every tree is proposed, e.g., adding a new node to a tree or removing one from it. Second, because marginalizing over the latent variable u for every sample is computationally expensive, DR-BART proposes using Gibbs or slice sampling to assign a u value to every data point. At every MCMC iteration, a Gibbs or slice sampler then updates each data point's u value.

DR-BART itself is an extension to BART (Chipman et al. 2010), which is a tree-based ensemble learning model for mean regression tasks. For BART, it has been acknowledged that its multi-tree structure makes it more robust against converging to local minima during training (Chipman et al. 2010).

Mutual information

MI was introduced in Shannon (1948), originally referring to it as "amount of information". It measures the amount of information that one random variable contains about another, without assuming a special relationship, e.g., linear, between the two variables. In other words, when an independent variable has a high MI value with the target variable, it does not imply a causal relationship, but is strongly informative about the target variable. For two probability distributions, MI can be defined as:

$$I(X;Y) = \int_{\mathcal{X}} \int_{\mathcal{Y}} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) dx\, dy \tag{1}$$

In practice, $p(x)$, $p(y)$, and $p(x,y)$ are all hard to estimate. Discretization of the data can be meaningful, as it allows the distributions to be approximated as frequency-based discrete distributions, and the integrals to be replaced by sums (Guyon and Elisseeff 2003).

$$I(X;Y) = \sum_{X} \sum_{Y} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) \tag{2}$$

Information bottleneck

The IB principle aims at finding "a tradeoff between compressing the representation [of data] and preserving meaningful information" (Tishby et al. 2000). It was introduced and formally defined by Tishby et al. (2000) as finding an optimal compression $\tilde{X}$ of the

[1] In fact, the implementation from Orlandi et al. (2021) does not use a sum-of-trees model for the log-variances, but a product-of-trees model for precisions, the reciprocals of variances: https://github.com/vittorioorlandi/drbart/

original data X to the relevant data Y, where β acts as a trade-off parameter, where a lower β prioritizes compression and a higher β relevance.

$$\arg\min_{\tilde{X}} \left(I(\tilde{X}; X) - \beta I(\tilde{X}; Y) \right) \tag{3}$$

Tishby et al. (2000) also introduce a top-down clustering algorithm for finding $\tilde{X}$, which yields a soft partitioning of X, meaning that data points can have probabilistic memberships to (several) clusters. Slonim and Tishby (1999) present a bottom-up agglomerative clustering approach which yields hard clusters, i.e., every data point can only belong to one cluster. Their algorithm finds a partition of X into m hard clusters such that $|\tilde{X}| = m$. In their agglomerative IB clustering algorithm, the β parameter from Eq. 3 is indirectly determined by the chosen number of clusters m. A lower amount of clusters favors compression, while a higher amount of clusters favors relevance.

Event log encoding for DR-BART

Pro3Log aims at learning probabilistic distributions for processing and waiting times from process event log data. A process event log $L = \{\sigma_1, \sigma_2, ..., \sigma_n\}$ is a multiset of traces. Each trace σ consists of a sequence of events $\sigma = \langle e_1, e_2, ..., e_m \rangle$. Each event e is a named tuple of event attributes $e = \langle a_1, a_2, ..., a_k \rangle$. More formally, there exists a bijection from every attribute name (also referred to as an attribute label) $l \in N$ to a positional index in the event tuple, $\phi : N \to I$, where $I = \{1, ..., k\}$.

Using event logs for learning probabilistic distributions for processing and waiting times with DR-BART poses the following challenges. At first, it is required that event logs contain information on the processing times of activities and preceding waiting times, which is often not directly captured in existing event logs. This challenge is addressed in Section "Durations in event log data". Secondly, DR-BART approximates the conditional probability density $p(y|x)$ for a fixed-size feature vector x. Since traces in event logs often have variable lengths, we show in Section "Prefix encoding" how the history of a trace can be compressed into a fixed-length set of attributes, in Section "Feature engineering" how contextual information, such as the time when a process is executed, can be encoded, and in Section "Inter-case features" how inter-case information can be added. Lastly, in Section "Encoding", we describe how the feature attributes are encoded into a feature vector required for DR-BART.

Durations in event log data

Pro3Log aims to learn probability distributions of processing times of activities or the waiting times preceding them, requiring the durations of their individual processing and waiting times. Event logs that implement a lifecycle model can provide exact information about processing and waiting times, e.g., the lifecycle models of XES (IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams 2023), the Business Process Analytics Format (BPAF) model (zur Muehlen and Swenson 2010), and CPEE (Mangler and Rinderle-Ma 2022) allow tracking state changes for the execution of activities. The XES lifecycle transaction model, for example, allows tracking the time when an activity has become ready, when the execution has started, been interrupted, continued, and finally completed or canceled. If an event log has implemented a lifecycle model, then activity processing times can be directly obtained

by, for example, calculating the time from the start of an activity until its completion or the waiting time by calculating the time from when an activity has become ready until it has been started (cf., e.g., temporal profiles as proposed in (Stertz et al. 2020) and implemented in PM4Py[2]).

However, existing event logs often do not implement lifecycles and hence do not differentiate between start and completion of activities, but only log one timestamp per activity. An example for such an event log can be seen in Table 1. The first row, for example, reflects the completion of activity `A` for case (instance) 1 at completion time `2024-10-31 07:00` by resource `Bob`. Note that we assume that the timestamps reflect completion times, but they could also reflect, e.g., the start times of the activities. Sometimes this information is available in the documentation of the event logs.

A simple approach to estimate durations is to calculate the duration from an event's timestamp to the timestamp of its preceding event (when the completion timestamp is logged) or its succeeding event (when the starting timestamp is logged) in the case.

The major problem of this approach is that when activities are processed in parallel, an activity might actually be preceded by another activity other than the one that has been completed in the trace before. Therefore, the calculated durations might underestimate the actual processing or waiting times. When there is no parallelism in the underlying process and only the completion of activities is tracked, then two problems remain. First, it is not possible to determine the duration of the first activity; Second, the obtained durations include processing as well as waiting times. Oftentimes, it may be desirable to decompose processing and waiting times as they might be influenced by different factors. E.g., the waiting time might be heavily influenced by the number of concurrently running process cases, while the actual processing time might rather be influenced by the conducting resource. Some works have addressed decomposing waiting and processing times from the overall duration. For example, Wombacher et al. (2011) make the assumption that a resource can only work on one activity at a time. They suggest defining an activity's processing time as the duration from the most recent timestamp before its completion, when the same resource finished another activity, to the activity's completion timestamp. Another approach by Fracca et al. (2022) addresses the decoupling via process simulation. Their approach aims to find, for each activity, an optimal global share value that divides its duration into waiting and processing times, based on process simulation by minimizing the event log distance from a simulated log to the original event log.

In *Pro3Log*, we use all possible information from the event logs, ideally, lifecycle events. In case only completion times are available, we calculate the differences between them in order to keep the approach simple.

Table 1 Example event log with completion times of activities

case	timestamp	label	resource
1	2024–10–31 07:00	A	Bob
1	2024–10–31 07:15	B	Alice
1	2024–10–31 08:30	B	Felix
2	2024–10–31 09:00	A	Alice
2	2024–10–31 09:15	B	Felix
1	2024–10–31 09:45	C	Bob

[2] https://pm4py-source.readthedocs.io/en/latest/pm4py.algo.discovery.temporal_profile.html, accessed 2015–08–05

Table 2 shows the event log from Table 1 with the durations obtained from an event's timestamp to the timestamp of its preceeding event in the case.

Prefix encoding

To encode the history of a running case and to reduce the sequence-based event log data to a fixed-sized input, required for DR-BART, prefix encoding techniques can be used (Verenich et al. 2019). Applicable to DR-BART are last m-states encoding and aggregation encoding. In the last m-states encoding, the m variable specifies the number of previous events of a case that are encoded. However, Verenich et al. (2019) note that the majority of publications choose $m = 1$, i.e., do only encode the most recent event and no previous events. We also select $m = 1$ in this work, as choosing a larger m would strongly increase the input size. We provide information on previous events by using aggregation encoding instead.

Aggregation encoding adds additional attributes to the event log that aggregate information about a case's previous events. Information about a numerical attribute can be aggregated, e.g., by adding a new attribute that represents the sum, average, minimum, or maximum value of the previous values. For categorical attributes, for each value, an additional column can be created with the number of occurrences of the categorical attribute value.

Feature engineering

Additionally, we propose applying feature engineering, i.e., deriving new feature attributes from existing features in the event log. In particular, we propose to apply feature engineering based on the timestamp attributes. As performances of human resources have been shown to differ over time (van der Aalst 2010), or as waiting times might also depend on the time, we add the `day of the week` and the `seconds in the day` attributes.

Inter-case features

Often, the processing and especially the waiting times of activities depend not only on intra-case features, such as a case's own history, but also on inter-case features, i.e., information about other cases. For example, the waiting time for an activity may be determined by the number of other cases that are also enqueued at that activity.

Several works have addressed encoding inter-case information as additional features (Senderovich et al. 2014, 2015, 2017, 2019).

Senderovich et al. (2017) propose to split the concurrently running cases into different case types. Their approach requires a practitioner to define a discrimination function, i.e., a function that maps a running case to one of m case types, and a derivation function that maps the information of cases that belong to a case type at a given timestamp to a feature space.

Table 2 Event log with durations

#	case	start timestamp	end timestamp	label	resource	duration in seconds
1	1	2024–10–31 07:00	2024–10–31 07:15	B	Alice	900
2	1	2024–10–31 07:15	2024–10–31 08:30	B	Felix	4500
3	1	2024–10–31 08:30	2024–10–31 09:45	C	Bob	4500
4	2	2024–10–31 09:00	2024–10–31 09:15	B	Felix	900

In their evaluation, they propose to use a count aggregation for the last activity label of each of the m case types as a derivation function. Furthermore, they propose different types of discrimination functions depending on their level of granularity. At the first level, only a single case type is chosen. The derivation function then counts the number of concurrently running cases, encoding this number as a single additional feature. Since a waiting time of an activity might rather depend on the number of concurrently waiting cases for that activity than on the overall number of running cases, they further propose a second-level and a third-level encoding, which take the last activity labels of the running cases into account. At the second level, m is chosen as the number of last activity labels. The derivation function then creates a count aggregation for each of the last activity labels. At a third level, m is chosen as the number of the last $w > 1$ activity labels in each case. The derivation function then creates count aggregations for each of these states as additional features.

In practice, the third-level approach can be prone to an explosion of the state space. Therefore, Senderovich et al. (2019) propose to encode as inter-case features the distances from a predefined distance metric to K nearest other cases. As a distance metric, they propose considering a temporal dimension, e.g., the distance between timestamps of individual events, and a control flow dimension, e.g., the string edit distance between activity labels.

Table 3 shows the event log from Table 1 with prefix encoding of the case activity and resource labels, feature engineering resulting in the `seconds in the day` and `day of the week` attributes, and a second-level instance encoding.

Encoding

To train and sample from DR-BART models, we need to encode the relevant event attributes for each event into a feature vector. As the case identifier and the two timestamp attributes do not contain relevant information for inference, we exclude them from the feature vectors. Instead, we consider all remaining attributes, except for the duration, as the input features and take the duration as the target feature. Note that the duration feature is only in a lifecycle event log clearly defined as either waiting or processing time.

Table 3 Event log with durations, prefix encoding, feature engineering and inter-instance features

	First Set of Attributes								
		Timestamps		Event Attributes		Prefix Encoding: Activity		
#	case	start	end	label	resource	A	B	C
1	1	2024–10–31 07:00	2024–10–31 07:15	B	Alice	1	0	0
2	1	2024–10–31 07:15	2024–10–31 08:30	B	Felix	1	1	0
3	1	2024–10–31 08:30	2024–10–31 09:45	C	Bob	1	2	0
4	2	2024–10–31 09:00	2024–10–31 09:15	B	Felix	1	0	0

	Second Set of Attributes								
	Prefix Encoding: Resource			Inter-Case Encoding			Feature Engineering		Target
#	Bob	Alice	Felix	A	B	C	seconds in the day	day of the week	duration in sec.
1	1	0	0	0	0	0	25200	4	900
2	1	1	0	0	0	0	26100	4	4500
3	1	1	1	0	0	0	30600	4	4500
4	0	1	1	0	1	0	32400	4	900

Moreover, DR-BART requires the input features and the target feature to be numerical. For categorical attributes, *Pro3Log* uses label encoding, where each category receives its own value starting from 1, while 0 is reserved for categories that have not been observed in the training data, but might occur during inference. Our target feature, the duration, is already a numeric variable, but since it can only take positive values, *Pro3Log* conducts a log transformation and trains on its logarithmic value. This ensures that during inference, sampled values will be strictly positive. For numerical stability, we additionally normalize and standardize the log-duration.

Training and inferring from DR-BART models

This section presents how *Pro3Log* learns and samples processing or waiting times. *Pro3Log* distinguishes between the basic approach in which one DR-BART model is trained on all events from an event log (Section "Basic *Pro3Log* approach"), and the MoE approach in which the events are split into several sub-logs on which individual expert DR-BART models are trained (Section "Mixture of experts approach"). While the basic approach addresses smaller event logs, the MoE approach is a novel approach that addresses scalability and complexity issues with larger event logs.

Basic *Pro3Log* approach

For the basic *Pro3Log* approach, the encoded attributes are used as feature attributes for training DR-BART. Training a DR-BART model uses an MCMC sampler that alternates between (i) slice sampling latent u_i for each training sample, and (ii) sequentially proposing a Metropolis-Hastings move for each tree and computing the acceptance probability using the likelihood evaluated with the u samples. In practice, MCMC iterations can become slow on large event logs. For one part, both slice sampling and evaluating the likelihood involve iterating over all training samples, resulting in linear computational complexity with the number of training samples. For the other part, both slice-sampling and evaluating the likelihood involve traversing the trees, which itself can also become slower with growing event logs. This is because larger event logs often additionally exhibit high-cardinality features, such as many activities or resources, which introduce additional complexity through increased potential interactions. As a result, the trees need to grow deeper or broader to effectively capture these interactions, meaning that traversing the trees will also take longer.

Orlandi et al. (2021) provide an implementation of their approach on GitHub[3], but for training with encoded event log attributes and on durations, we noticed issues with numerical stability. These numerical instabilities arise primarily because DR-BART can learn degenerate normal distributions. In some situations, such degenerate distributions are justified, for example, when DR-BART identifies automated activities that always take a constant amount of time and therefore exhibit no observable variance. However, such distributions then cause numerical issues. First of all, such degenerate probability distributions may cause the likelihood calculated at every MCMC iteration to become infinite, which leads to problems at the Metropolis-Hastings acceptance step. Moreover, such distributions can cause the slice sampler to become trapped in an effectively infinite loop, as the high-probability region becomes excessively narrow and difficult to

[3] https://github.com/vittorioorlandi/drbart/

locate. Conversely, DR-BART can also temporarily learn normal distributions that assign extremely low likelihoods to individual training samples. This may cause the joint likelihood across all training samples to become zero (or the log likelihood to become negative infinity, respectively), which in turn leads to problems at the Metropolis-Hastings acceptance step. Therefore, we have refined their DR-BART implementation for better numerical stability, which is publicly available on GitHub as well.[4]

Inferring the full conditional density

Given a feature vector x, i.e., encoded attributes, DR-BART can infer the full conditional density $p(y|x)$ for the target attribute y, i.e., the duration. This is done by obtaining the conditional densities for each MCMC iteration $p(y|x, \theta^t)$ and averaging them. Then, for each MCMC iteration, the conditional density is obtained by marginalizing over the latent variable u. In practice, this can be done by keeping track of the splits of the latent space induced by u. Then, for each u-interval, the trees can be traversed and their leaves summed up to obtain a mean and a variance parameter for a normal distribution. This normal distribution then makes up one component in a Gaussian mixture model, weighted by the u-interval's length.

Sampling from DR-BART models

Having the full conditional density at hand allows, for example, to access the exact probability density for a duration. However, for many applications, such as business process simulation, one would rather want to draw a single sample from the conditional density.

This can be achieved computationally cheaply by first randomly selecting a (post-burn-in) MCMC iteration $t \sim \text{Uniform}\{1, ..., T\}$, where T is the number of post-burn-in MCMC iterations and randomly sampling a value of the latent variable $u \sim \mathcal{U}(0, 1)$. With the sampled u, the trees from the MCMC iteration θ^t can then be traversed and their leaves summed to obtain a mean and a variance parameter for a normal distribution. From this normal distribution, a sample can then be drawn. Because in *Pro3Log* the durations are transformed into log-space for training, the sampled value must be inverse-transformed by exponentiation to obtain a duration.

Mixture of experts approach

Training the standard approach on large event logs can be computationally costly. Additionally, to capture the complexities of large event logs, such as those arising from numerous activities and resources, DR-BART trees need to be large, which further detracts from training and inference performance.

For providing faster training, we present our MoE approach that aims at splitting the event log into smaller sub-logs (clusters) on which individual DR-BART models are then trained. During inference, the same splitting criterion is applied to route a feature vector to the appropriate DR-BART 'expert' model. Therefore, we refer to this approach as the MoE approach.

Ideally, every cluster should be (i) conditionally sufficient for prediction, so that no information from other clusters is required to make an accurate prediction, and (ii) the clustering should reduce per-expert model complexity by partitioning the data into

[4] https://github.com/ltsstar/drbart/

regions where the target function is simpler. In information-theoretic terms, the clustering should (i) preserve the mutual information between the input and the target, i.e., maximize $I(\tilde{X};Y)$, and (ii) reduce the per-expert complexity, i.e., $I(\tilde{X};X)$, which is equal to Eq. 3.

Therefore, *Pro3Log* MoE approach uses the agglomerative IB algorithm that finds (up to) n clusters that minimize the loss of predictive information by maximizing the MI of the clusters to the target variable $I(\tilde{X};Y)$. The agglomerative IB approach from Slonim and Tishby (1999) addresses only one feature variable, while we are concerned with multiple feature variables. Since the number of possible partitions across all feature variables is often intractably huge, *Pro3Log* limits the partitioning only accross individual feature variables, which has also been proposed and investigated in Slonim et al. (2001). In particular, we use to agglomerative IB algorithm on each feature variable X_j to find a partitioning $\tilde{X}_j$ that minimizes $I(\tilde{X}_j;Y)$ and select the partitioning with the highest mutual information $\arg\max_j \left(I(\tilde{X}_j;Y)\right)$. Thereby, our MoE approach uses a single feature variable for partitioning the event log into (up to) n clusters.

Training the MoE approach can be outlined as follows. First, a predefined number of n clusters is defined. All continuous attributes (including the duration) are discretized into m bins. Then, the agglomerative IB algorithm is applied to find up to n clusters from the feature variables' categories that maximize the MI with the discretized duration attribute. Eventually, the attribute with the highest MI with the target data on its clusters is selected as the splitting criterion for the event log data. Note that some categorical feature variables may have fewer than n categories. For these variables, we calculate their MI with their existing categories as clusters. If such a variable has the highest MI across all variables, we select its categories as the splitting criterion, which means that in such a case, the number of clusters n is the number of the variables categories. Finally, the event log data is split according to the selected splitting criterion, and for each of the sub-event logs, individual DR-BART models are trained.

During inference, a sample is routed to the respective DR-BART model by identifying the cluster from the value of the sample's splitting variable. It might happen that this value matches with no cluster. When the splitting variable is continuous, then that might be due to the value of the sample's splitting variable being larger than the largest observed value, or smaller than the smallest observed value in the training data. In these cases, the cluster with the largest or smallest range, respectively, is picked. When the splitting variable is categorical, a new category might be encountered (e.g., a new activity or resource). In such a case, we select a random cluster.

Evaluation

In this section, we evaluate *Pro3Log* by applying it in a business process simulation model to sample processing or waiting times of activities. First, we describe the five event logs on which we evaluated *Pro3Log*. We then describe how we set up the business process simulation model and train the basic and MoE versions of *Pro3Log*. Finally, we describe the metrics applied to evaluate the simulation results. We provide a publicly available implementation for training the standard and MoE *Pro3Log* approach, and the presented evaluation approach on GitHub[5].

[5] https://github.com/ltsstar/TaskExecutionTimeMining/

Table 4 Event log properties

	Cases	Events	Variants	Event labels	Resources	Events per case /Duration (Mean ± Std.Dev.)
Artificial	1802	16209	1	3	5	9.00 (0.00) / 12.18 (5.49) hours
PCR	6166	117703	1213	8	-	19.09 (3.37) / 5.52 (7.74) hours
BPIC-17	31509	1202267	15930	26	159	38.16 (16.72) / 21.9 (13.17) days
BPIC-19	251734	1595923	11973	42	628	6.34 (13.07) / 71.52 (152.78) days
Helpdesk	4580	21348	226	14	22	4.66 (1.18) / 40.86 (8.39) days

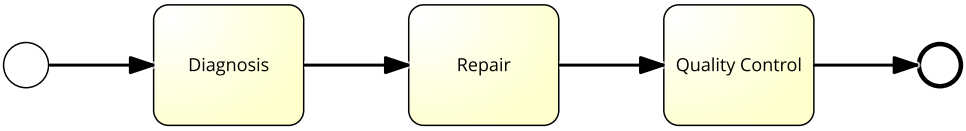

Fig. 2 Process model of the artificial process

Evaluation datasets

We use one artificial and four real-world data sets to train and evaluate the basic and MoE *Pro3Log* approaches. The event logs are chosen to cover a range of domains and properties, which are depicted in Table 4. We perform a train/test split on all event logs, dividing process traces so that 80% of cases form the training data and 20% constitute the test data. In the following, we describe the data sets in more detail.

Artificial event log

The artificial (AR)[6] event log describes a sequential process with three activities, resembling a repair shop in the manufacturing domain. The corresponding process model is depicted in Fig. 2.

While the process has a simple control flow, for the processing times of the individual activities, different degrees of uncertainty are involved, depending on different parameters. First, each activity has its base duration, which is sampled from log-normal distributions, where the mean and the variance parameters vary across the activities. Then, depending on the resource that conducts the activity, an offset is added to the base duration. This offset depends in part on the resource itself. The process has one type of resource that, in some cases, takes a break during the processing which is not explicitly tracked in the event log, resembling, e.g., getting a coffee, and another type of resource that conducts the activities faster in the morning than in the afternoon, resembling, e.g., an early bird person. The offset also depends on the interplay between the conducting resource and a resource that has previously conducted an activity. This resembles, e.g., that two resources are not good at cooperating, resulting, e.g., in the quality control activity taking much longer when it is conducted by a particular resource and when another specific resource has done the preceding repair activity. Because the data set is artificial, we can compare the performance of our approaches to the optimal, underlying probabilistic model.

[6] AR: https://github.com/ltsstar/TaskExecutionTimeMining/blob/main/data/artificial_event_log_2.xes

Polymerase chain reaction (PCR) event log

The Polymerase Chain Reaction (PCR)[7] event log is a real-world event log from a coronavirus testing laboratory that has conducted polymerase chain reaction tests. The event log captures a process that was orchestrated by a workflow engine based on an imperative process model. Parts of the activities are rather complex and require human resources, while others are automated and simple, such as simple timeout activities. The PCR event log is the only event log that is missing a resource identifier, meaning that our approaches must rely on other features.

BPIC-2017 event log

The BPIC-17[8] event log is a real-world data set from the financial domain. It is a loan application process from a Dutch bank and has been widely investigated in the Business Process Intelligence Competition (BPIC) 2017. It is a life cycle event log where the individual activities are often suspended and resumed at a later point.

BPIC-2019 event log

The BPIC-19[9] event log reflects a purchase-to-order process from a Dutch company in the field of coatings and paints. It has the largest number of cases and events, as well as the most resources. Additionally, it has the highest variance in case durations.

Helpdesk event log

The Helpdesk[10] event log is extracted from a ticketing management process of the helpdesk of an Italian software company over a timespan of approximately 4 years. Compared to the other event logs, the Helpdesk data set has the second-lowest number of cases and events. In comparison with the BPIC-2019 event log, the Helpdesk event log has two orders of magnitude fewer cases and events, but has only one order of magnitude fewer resources and only a third of the event labels of the BPIC-2019 event log, resulting in a lower feature-to-sample ratio than the BPIC-2019 event log.

Training probabilistic models

At first, the event logs are encoded as described in Section "Event log encoding for DR-BART". The AR, BPIC-17, and PCR event logs are life-cycle event logs for which the processing and waiting times can be extracted directly from the event log. BPIC-19 and Helpdesk only track the completion times of activities. Therefore, we estimate the duration as the timespan between the completion of an event and the completion of the preceding event from the same case.

Hyperparameter selection and DR-BART training

For training the *Pro3Log* standard approach, several hyperparameters of DR-BART must be selected. First, DR-BART itself consists of a set of hyperparameters that function as regularizing priors. These are primarily the α and β parameters for regularizing the shape of the trees. Orlandi et al. (2021) also define the hyperparameter a_0, which

[7] PCR: https://doi.org/10.5281/zenodo0.11617408

[8] BPIC-17: https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b

[9] BPIC-19: https://doi.org/10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1

[10] Helpdesk: https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb

parameterizes the prior distribution for the leaf parameters in the log-variance trees (which approximates $\mathcal{N}(0, (a_0 m_{\text{var}})^{-1})$) and the hyperparameter k, which parameterizes the prior distribution for the leaf parameters in the mean trees ($\mathcal{N}(0, \sigma_\mu^2)$, where $\sigma_\mu = \frac{0.5}{k\sqrt{m_{\text{mean}}}}$). As tuning these hyperparameters is computationally intractable, we use the default parameters as proposed in Orlandi et al. (2021) and Chipman et al. (2010), namely: $\alpha = 0.95$, $\beta = 2$, $a_0 = 1$, and $k = 2$. For the same reason, we choose the default number of mean and variance trees as $m_{\text{mean}} = 200$, $m_{\text{var}} = 100$.

For training DR-BART models, the number of MCMC iterations must be chosen. This number is composed of a number of burn-in iterations that are discarded, the number of post-burn-in iterations that are retained, and optionally a thinning parameter that defines whether only every k-th sample from the post-burn-in iterations is retained. We conducted initial tests and noticed that the model's likelihoods most often had converged after a few thousand iterations. Therefore, we set the total number of MCMC iterations for all DR-BART models to 10,000 with 7500 burn-in iterations and 2500 post-burn-in iterations, of which every 50-th sample was kept. For training the standard approach on the BPIC-2017 and BPIC-2019 event logs, training for 10,000 iterations proved to be computationally infeasible. Therefore, we reduced the number of MCMC iterations for these event logs to 1000 with 750 burn-in iterations and 250 post-burn-in iterations of which every 5-th sample was kept (see Table 5).

For the *Pro3Log* MoE approach, the number of clusters must also be selected. We decided to select $n = 32$ clusters for all data sets. A MCMC iteration for the individual expert models is typically significantly faster than an iteration in the standard approach, as the training dataset size for each expert model is typically substantially smaller.

It should be noted that selecting a large number of post-burn-in iterations to retain and a large number of MoE clusters can quickly lead to a large model size. While, with the selected hyperparameters, the individual trees are often small, i.e. mostly having less than 1000 parameters, the number of clusters, of trees, and of post-burn-in iterations scales the total parameter size heavily: e.g., having 300 trees in total and keeping 50 MCMC iterations for each of the 32 experts makes up already 480,000 trees to save. While this number does not cause problems with recent hardware, storing many more MCMC iterations, therefore, may quickly become impractical.

Feature attribute selection

To test the impact of the selected features on the performance of *Pro3Log*, we evaluate the approach using different combinations of feature attributes. We begin testing combinations that select only the current activity or resource, such as traditional approaches in business process simulation have done (López-Pintado et al. 2024; Rozinat et al. 2009).

Table 5 MCMC iterations

	Numerical Stable DR-BART			MoE		
	Burn-In	Post-Burn-In		Burn-In	Post-Burn-In	
		Kept samples	Thinning		Kept samples	Thinning
Artificial	7500	50	50	7500	50	50
PCR	7500	50	50	7500	50	50
BPIC-17	750	50	5	7500	50	50
BPIC-19	750	50	5	7500	50	50
Helpdesk	7500	50	50	7500	50	50

We then test the impact of additional features, such as the seconds in a day or the day of the week features. Finally, we test for a full combination of features, i.e., count aggregations for the previous activities and resources, the seconds in the day, and the day of the week features, as well as inter-case encodings. As inter-case encodings, we use the second-level encoding from Senderovich et al. (2017), i.e., encoding the last states of other running cases as count aggregations, which we denote with `ii1`, and the third-level encoding with $m = 3$, i.e., encoding the last three states of other running cases as count aggregations, which we denote as `ii3`.

Comparison

Existing works in the field of business process simulation and predictive process monitoring have addressed learning the processing and waiting times of process activities (cf. Section "Related work"). Many works have only trained regression models that can be used to make point estimations, and are therefore unable to capture the uncertainties inherent to processing and waiting times. A few works have explicitly addressed the learning of probability distributions, while other regression models can be easily adapted to predict distributions. We implement three baseline approaches based on existing methods, i.e., PIX, Quantile Regression, and Gaussian LSTM, and compare them against our `Pro3Log` approach.

PIX framework

Traditional data-driven business process simulation approaches propose fitting parametric probability distributions, such as normal distributions, to the processing and waiting times of each activity (Rozinat et al. 2009). The PIX framework[11] offers a recent implementation for fitting parametric probability distributions to processing or waiting times. In its implementation, it first fits several parametric probability distributions, e.g., a uniform, (log-)normal, triangular, exponential, or gamma distribution, on the given data and then selects the distribution which achieves the highest likelihood on the training data. In López-Pintado et al. (2024), the PIX framework is used to fit i) probability distributions for the processing times of each activity, and ii) probability distributions for each activity and resource pair, because they assumed that the resources that conduct the activities affect their shape. The authors demonstrate that, in real-world event logs, training (conditional) probability distributions for each activity-resource pair and applying their model in business process simulation can contribute to better simulation results. Hence, the approach is called *resource-differentiated* approach. We also use the PIX framework twice as a baseline approach: first, we follow a traditional approach and use the PIX framework to train a probability distribution for the processing or waiting times associated with each activity label; secondly, we utilize the PIX framework to train resource-differentiated probability distributions. Since some activity-resource combinations are rare, which can lead to non-representative or degenerate probability distributions, we require a pair to consist of at least 15 samples in the event log. When fewer samples for a pair are recorded, we resort to a probability distribution that is fitted to the durations for the respective activity label.

[11] https://github.com/AutomatedProcessImprovement/pix-framework

Quantile regression

Several regression models have been used to make point-estimations for remaining time predictions (Verenich et al. 2019). Similar to DR-BART, many of these approaches have used tree-based ensemble methods, e.g., XGBoost (Verenich et al. 2019). Current works typically train their regression models to optimize either towards the median by minimizing the mean absolute error during training, or towards the mean by minimizing the mean squared error. A loss function can also be adopted to train on quantiles, referred to as quantile regression (Koenker and Bassett 1978).

We use quantile regression as a baseline approach to train on the $\tau = 0.5$ quantile, i.e., the median, and the other model on $\tau = 0.8413$, which is approximately the quantile of the first upper standard deviation. During inference, the predictions of both models are used as parameters of a normal distribution, where the first is used as the mean parameter, and the margin between the predictions of the median and the first upper standard deviation is used as the standard deviation. The resulting normal distribution is then used to realize a duration during process simulation.

As quantile regression models, we use a tree-based ensemble model, namely histogram-based gradient boosting regression trees, and use the implementation from the sklearn library with its default parameters.[12]

Gaussian LSTM

Sequential models, such as LSTM neural networks, have been shown to perform well in predicting remaining times of business processes (Verenich et al. 2019). A primary advantage of sequential models is that they do not require feature encoding approaches, but instead work directly on sequential data with varying sequence lengths. Sequential models can not only be used to conduct a mean regression, but also for a heteroscedastic regression, e.g., predicting the mean and the variance of a normal distribution. Such models have been applied to predict remaining time intervals of business process cases (Kunkler et al. 2025; Weytjens and Weerdt 2022). Since this work focuses on time predictions for activities, we train as a baseline approach Gaussian LSTMs to predict the mean and standard deviation of a normal distribution for processing and waiting times. For training our Gaussian LSTMs, we use the event label, the resource, and the seconds in the day and the day of the week features for all events in the trace prefix as input. We use two-layer LSTMs with a hidden size of 128, use feature embedding for the categorical variables (the event label and the resource), and train each model for 200 epochs. During simulation, the mean and standard deviation are predicted from the prefix trace and used as parameters of a normal distribution, from which a sample is then drawn.

Business process simulator

We compare our approaches with the baseline approaches in a business process simulation model to simulate process cycle times. Since our focus is on evaluating processing and waiting times, our simulator replays the events of process cases and uses the respective probabilistic models to sample durations for processing and waiting times. Similarly, the resource allocations and the case start times are replayed from the event log. For all but the PCR data set, the simulator obtains a case cycle time by executing the activities

[12] https://scikit-learn.org/1.7/modules/generated/sklearn.ensemble.HistGradientBoostingRegressor.html

sequentially, i.e., the waiting and processing times are summed up. For the PCR data set, an imperative process model exists, enabling that activities are also executed in parallel.

Monte Carlo sampling

Deriving a process cycle time probability distribution analytically is, in most cases, intractable. Therefore, we approximate the probability distribution of process cycle times via Monte Carlo (MC) sampling.

While drawing samples for a single case is computationally cheap, drawing many MC samples for every case in the test data sets can become computationally difficult. To find a good tradeoff between achieving precision from sampling sufficiently many times and computational load, we choose to draw 1000 samples for each case across all event logs.

Evaluation metrics

We evaluate how the sampled process case cycle times align with the true cycle times by using two common proper scoring rules, which "assess the quality of probabilistic forecasts, by assigning a numerical score based on the predictive distribution and on the event or value that materializes" (Gneiting and Raftery 2007). Note that other works that have addressed the evaluation of BPS models have proposed to draw only a single sample for each case and then to compare the distribution of the samples from all cases with the true cycle time distribution of all cases (see e.g., Chapela-Campa et al. (2025)). While such an evaluation approach is well-suited to evaluate whether the simulated distribution of process case cycle times aligns with the true distribution of cycle times, it does not allow for evaluating the precision on a per-case basis. To illustrate this, consider two cases with different cycle times. The BPS model would achieve a perfect result when it either samples for both cases the true cycle time, or when it samples for the first case the cycle time of the second case, and vice versa. To facilitate a per-case evaluation, we employ two proper scoring rules to assess the drawn process cycle time samples for each case against the ground-truth process cycle time for that case. Then we aggregate the per-case results across all cases to a and aggregate the per-case results into a metric score.

Average log-likelihood

The first metric we use is the average log-likelihood, where higher average log-likelihood values are desirable. We obtain the average log-likelihood by estimating a continuous probability distribution via kernel density estimation for every case's samples. Then we take the probability distribution's logarithmic probability density at the ground truth value.

The probability density of a case i for its true cycle time x_i on the sampled cycle times $X_i = (x_1, ..., x_n)$ is obtained via:

$$\hat{f}(X_i, x_i) = \frac{1}{n} \sum_{j=1}^{n} \mathcal{N}(X_{ij} - x_i, h) \tag{4}$$

Note that the probability density depends on the bandwidth parameter h. We use Silverman's rule of thumb Silverman (1971) to estimate the bandwidth parameter. It should be noted that Silverman's rule of thumb is based on the assumption that the underlying

data is normally distributed. When the data is not normally distributed, Silverman's rule of thumb can lead to oversmoothing in the probability distribution. However, in our experiments, extremely low probability density at the ground truth often caused numerical instability, making oversmoothing preferable to undersmoothing for more stable and reliable kernel density estimates. Furthermore, advanced techniques for bandwidth tuning, e.g., via cross-validation, can become computationally too expensive. Therefore, we decided to use Silverman's rule of thumb. The average log-likelihood is defined as:

$$\mathcal{LL}(X, x) = \frac{1}{n} \sum_{i=1}^{n} \log\left(\hat{f}(X_i, x_i)\right) \tag{5}$$

While the log-likelihood does not provide information about calibration at a per-sample level, well-calibrated models tend to achieve higher average log-likelihood scores across all test samples. A characteristic of the log-likelihood score is that it has a high sensitivity to outliers, i.e., ground truth values that lie in low-probability regions of the estimated probability distribution achieve extremely low log-likelihood scores. In practice, this sensitivity can lead to numerical instabilities since the log likelihood diverges to negative infinity for such outliers. Consequently, we were unable to report log-likelihood values for every model across the datasets. Therefore, we additionally compute the continuous ranked probability score, offering a numerically stable alternative for assessing the performance of the probabilistic models.

Continuous ranked probability score

The second metric we use is the average continuous ranked probability score (CRPS), where a lower CRPS is desirable. Unlike the average log-likelihood metric, the CRPS metric is sensitive to the distance of the predicted case cycle times to the true cycle time. We adapt the notation of Hersbach (2000) and define the CRPS for one case i as a distance between the empirical cumulative density function $F_{X_i}(x)$ of the sampled cycle times and $F_{x_i}(x)$, a shifted Heaviside function, shifted by the true cycle time x_i.

$$\begin{aligned} CRPS(X_i, x_i) &:= \int_{-\infty}^{\infty} [F_{X_i}(u) - F_{x_i}(u)]^2 du \\ F_{X_i}(x) &:= \frac{1}{n} \sum_{j=1}^{n} \mathbf{1}(X_i \leq x) \\ F_{x_i}(x) &:= \begin{cases} 0 & \text{if } x_a > x \\ 1 & \text{if } x_a \leq x \end{cases} \end{aligned} \tag{6}$$

While the log-likelihood score of a sample is heavily influenced by samples close to the true duration, the CRPS metric is more sensitive to the entire distribution of samples. Furthermore, the CRPS metric assesses a model's calibration on a per-sample level.

Results

In this section we present and review the results based on the two metrics across the different event logs. In Section "Pro3Log: the basic variant and the MoE variant" we compare the results between *ProLog*'s basic and MoE version. In Section "Pro3Log: feature attribute combinations" we assess the impact of different feature attribute combinations

and compare the performance of *Pro3Log* with the baseline approaches in Section "Pro-3Log and the baseline approaches".

Pro3Log: the basic variant and the MoE variant

The results for the different feature combinations of Pro3Log's basic and MoE versions are presented in Table 7 for the PCR event log and in Table 6 for the other event logs. Due to numerical instabilities arising from extremely low likelihoods of specific process cycle time samples in some event logs when using certain models, we were sometimes unable to compute the average log-likelihood scores (denoted with - in Tables 6 and 7). These numerical instabilities occurred twice for the MoE approach on the BPIC-17 event log, 16 times in the BPIC-19 event log, and twice for the MoE approach on the PCR event log. Since the CRPS metric provides a robust alternative, this does not undermine the validity of the results.

When comparing the results of the basic variants of *Pro3Log* with the MoE variants, we can see that on three event logs (AR, BPIC-17, and PCR), at least one MoE variant exists that achieves better results on both metrics than all basic variants. Only on the Heldesk data set, basic variants can outperform all MoE variants. On the BPIC-19 data set, a basic variant achieved the best average log-likelihood, but a MoE approach yielded the best CRPS score.

While the average log-likelihood metric appears to be affected by outliers as it exhibits a strong variance across the variants, the CRPS metrics show a rather constant trend: for the CRPS metric, the best feature combination for the MoE variants outperforms the best feature combination for the basic variant counterpart on every dataset except the Helpdesk dataset. The good performance of the MoE variants on the AR, BPIC-17, BPIC-19, and PCR event logs might stem from the fact that the clustering successfully reduced the overall complexity, allowing individual experts to focus on learning important interactions. Conversely, the poor performance of the MoE variants on the Helpdesk event log may be due to poor clustering, such as high inter-cluster dependencies, or an overall low complexity present in the log. Additionally, the Helpdesk event log contains the second-lowest number of events, exceeding only the AR event log in this regard, while it exhibits a substantially greater diversity of event labels and resources compared to the AR dataset. This indicates that the Helpdesk event log provides relatively limited data for learning complex relationships and complex, e.g., multi-modal, probability distributions.

Overall, the results suggest that the *Pro3Log* basic variant should only be chosen over the MoE variant when the processing or waiting times exhibit low complexity or the training data is sparse.

Pro3Log: feature attribute combinations

The synthetic AR event log was the only event log for which the relevant feature combinations are known (i.e., the activity, resource, resource count, and the seconds in day attributes): using them achieved, in fact, yielded the best results on the CRPS metric for the MoE variant of *Pro3Log*.

However, in practical applications, the relevant feature combinations are often not known, and testing multiple feature attribute combinations for *Pro3Log* may not always be feasible. Our results show that across all event logs, using the event label alone often

Table 6 Results for the event logs across the different event logs for the input features

model	input features								AR				Helpdesk			
									basic approach		MoE approach		basic approach		MoE approach	
	a	r	ac	rc	s	d	ii1	ii3	LL	CRPS	LL	CRPS	LL	CRPS	LL	CRPS
DRB	x	-	-	-	-	-	-	-	−0.01	1.25E4	−0.36	1.02E4	**−4.08**	6.74E5	−4.17	7.08E5
DRB	-	x	-	-	-	-	-	-	−0.27	2.01E4	0.06	1.02E4	−4.67	1.35E6	−4.50	9.58E5
DRB	x	x	-	-	-	-	-	-	−0.04	1.23E4	0.17	9.83E3	−4.12	6.78E5	−4.19	7.47E5
DRB	x	x	-	-	x	-	-	-	−0.04	1.31E4	−0.45	9.73E3	−4.24	7.68E5	−4.63	9.75E5
DRB	x	x	x	-	x	-	-	-	−0.48	1.13E4	−0.41	9.68E3	−4.13	***6.36E5***	−4.67	1.09E6
DRB	x	x	-	x	x	-	-	-	−0.15	1.19E4	0.13	***8.78E3***	−4.21	7.41E5	−4.86	1.23E6
DRB	x	x	x	x	x	-	-	-	−0.24	1.14E4	***0.40***	8.83E3	−4.56	9.56E5	−4.89	1.31E6
DRB	x	x	-	-	x	x	-	-	−0.07	1.11E4	−1.13	9.79E3	−4.24	8.19E5	−4.65	9.95E5
DRB	x	x	x	x	x	x	-	-	−0.10	1.47E4	0.13	8.93E3	−4.28	7.56E5	−4.91	1.32E6
DRB	x	x	x	x	x	x	x	-	−0.17	1.02E4	0.00	8.99E3	−4.29	8.61E5	−5.05	1.43E6
DRB	x	x	x	x	x	x	-	x	−0.78	1.02E4	0.32	9.01E3	−4.39	7.85E5	−5.29	1.58E6

model	input features								BPIC-17				BPIC-19			
									basic approach		MoE approach		basic approach		MoE approach	
	a	r	ac	rc	s	d	ii1	ii3	LL	CRPS	LL	CRPS	LL	CRPS	LL	CRPS
DRB	x	-	-	-	-	-	-	-	−4.91	1.16E6	−3.94	7.71E5	***−14.03***	2.24E7	−52.27	2.70E6
DRB	-	x	-	-	-	-	-	-	−5.59	1.74E6	−4.08	***7.45E5***	−20.21	4.07E7	−82.39	2.32E6
DRB	x	x	-	-	-	-	-	-	−670.53	8.43E5	***−3.89***	7.52E5	-	2.77E7	−62.95	2.21E6
DRB	x	x	-	-	x	-	-	-	−4.64	9.86E5	−3.91	7.76E5	-	2.86E6	−147.25	2.57E6
DRB	x	x	x	-	x	-	-	-	−5.42	1.48E6	−4.52	7.64E5	-	3.60E6	-	2.09E6
DRB	x	x	-	x	x	-	-	-	−6.04	2.18E6	−3.96	7.96E5	-	4.52E6	-	1.93E6
DRB	x	x	x	x	x	-	-	-	−6.20	1.59E6	−6.49	9.38E5	-	9.14E6	-	1.94E6
DRB	x	x	-	-	x	x	-	-	−3119.13	9.08E5	−4.01	8.58E5	−102.22	2.81E6	-	2.34E6
DRB	x	x	x	x	x	x	-	-	−121.69	9.36E5	−4302.09	1.20E6	-	9.47E6	-	4.05E6
DRB	x	x	x	x	x	x	x	-	−6.51	1.37E6	-	1.13E6	-	9.63E6	-	***1.90E6***
DRB	x	x	x	x	x	x	-	x	−6.83	4.10E6	-	8.04E5	-	1.03E7	-	1.96E6

(a : activity label, r : resource, ac : count aggregation for the activity label, rc : count aggregation for the resource label, s : seconds in the day, d : day of the week, ii1 : second-level inter-case encoding, ii3 : third level inter-case encoding). The bold values denote the best *Pro3Log* results, the *italic* values represent the overall best results

Table 7 Results for the PCR event log

model	input features								PCR single model		MoE	
	a	r	ac	rc	s	d	ii1	ii3	LL	CRPS	LL	CRPS
DRB	x	-	-	-	-	-	-	-	–0.74	4.02E4	***1.08***	***9.04E3***
DRB	x	-	-	-	x	-	-	-	–10.24	2.13E4	–41.67	1.81E4
DRB	x	-	x	-	x	-	-	-	–5.02	1.92E4	–0.38	2.30E4
DRB	x	-	-	-	x	x	-	-	–45.09	2.26E4	–3.48	1.56E4
DRB	x	-	x	-	x	x	x	-	–257.75	9.96E3	-	1.85E4
DRB	x	-	x	-	x	x	-	x	–642.55	1.20E4	-	1.57E4

achieves the best overall log-likelihood score. Examining the CRPS metric, using the resource label in addition to the activity label yielded improved results compared to the activity label-only encoding for the MoE variant on all datasets, except for the Helpdesk dataset. Using more complex encodings, such as prefix encodings or inter-case encodings, could contribute only to an improved CRPS score in the MoE variant for the AR and BPIC-19 datasets.

Consequently, practitioners should select only those features for *Pro3Log* that are expected to impact the duration. In cases where the influence of a feature is unknown, increasing the number of training iterations may be beneficial to better capture potential relationships.

Pro3Log and the baseline approaches

The results for baseline approaches are shown in Table 8. Similar to the *Pro3Log* evaluations, the computation of the average log-likelihood scores caused numerical instabilities for the quantile regression approach for one feature combination on the BPIC-17 event log, for 10 feature combinations on the BPIC-19 event log, and for 4 feature combinations on the PCR event log; additionally, for the LSTM-based approach we were unable to compute the average log-likelihood score on the AR event log.

By comparing the baseline approaches with one another, we can see that the PIX framework outperforms the other baseline approaches in most cases. Interestingly, considering the resources in the PIX framework did not always outperform the activity-only variant. Comparing the results from *Pro3Log* with those from the baseline approaches, we can see that on the AR, BPIC-17, and PCR event logs, a *Pro3Log* MoE variant achieved the best results for both metrics. Considering only the CRPS metric, *Pro3Log*'s basic variant could achieve the best score on the Helpdesk data set and the MoE variant on the BPIC-19 event log. Therefore, on every event log, at least one *Pro3Log* variant could outperform all baseline approaches on the CRPS metric.

Comparing *Pro3Log* with only the activity-resource pair as input features with the activity-resource variants from the baseline approaches, we can see that on the BPIC-17 event log, the MoE variant of *Pro3Log* could overall achieve the best results on both metrics, while on the Helpdesk and BPIC-19, the PIX framework could achieve better results, while on the AR event log the results were comparable. This shows that *Pro3Log*'s advantage may lie in utilizing more features, as on the AR and BPIC-19 datasets, *Pro3Log* could outperform PIX in the CRPS metric when more features were used.

The advantage of *Pro3Log* lies in its ability to consider more features and learn the interactions between them. While the AR event log demonstrates that *Pro3Log* can, in

Table 8 Results for the pix framework, the quantile regression (QR) approach, the LSTM-based approach, and for the artificial dataset, the results from using the data generating probability distribution. As the PCR dataset does not have resources, the results are without the resource column

model	input features								AR		Helpdesk		BPIC-17		BPIC-19		PCR	
	a	r	ac	rc	s	d	ii1	ii3	LL	CRPS	LL	CRPS	LL	CRPS	LL	CRPS	LL	CRPS
PIX	x	-	-	-	-	-	-	-	−1.47	1.01E4	−4.03	6.67E5	**−4.31**	**8.17E5**	−23.02	2.32E6	**−0.49**	1.66E4
PIX	x	x	-	-	-	-	-	-	**−0.10**	**9.81E3**	*−4.00*	**6.49E5**	−4.43	8.25E5	−42.61	**2.05E6**	n.a.	n.a.
model	**input features**								**AR**		**Helpdesk**		**BPIC-17**		**BPIC-19**		**PCR**	
	a	**r**	**ac**	**rc**	**s**	**d**	**ii1**	**ii3**	**LL**	**CRPS**	**LL**	**CRPS**	**LL**	**CRPS**	**LL**	**CRPS**	**LL**	**CRPS**
QR	x	-	-	-	-	-	-	-	−5.09	1.02E4	−4.27	8.95E5	−6.17	1.04E6	−101.20	3.07E6	−23.44	1.49E4
QR	-	x	-	-	-	-	-	-	−0.75	1.12E4	−5.07	2.06E6	−15.95	1.25E6	-	3.13E6	n.a.	n.a.
QR	x	x	-	-	-	-	-	-	−1.15	1.01E4	−4.36	8.60E5	−6.32	1.00E6	-	2.69E6	n.a.	n.a.
QR	x	x	-	-	x	-	-	-	−767.04	1.27E4	−5.58	9.55E5	−6.04	1.01E6	-	2.72E6	−10540.65	1.60E4
QR	x	x	x	-	x	-	-	-	−717.16	1.27E4	−4.20	8.38E5	−6.68	1.64E6	-	2.55E6	-	2.07E4
QR	x	x	-	x	x	-	-	-	−393.40	1.23E4	−5.58	9.55E5	−6.07	1.32E6	-	2.82E6	n.a.	n.a.
QR	x	x	x	x	x	-	-	-	−315.55	1.23E4	−13.72	1.05E6	-	1.73E6	-	2.42E6	n.a.	n.a.
QR	x	x	-	-	x	x	-	-	−334.55	1.15E4	−4.75	1.02E6	−6.03	1.05E6	-	2.78E6	-	1.44E4
QR	x	x	x	x	x	x	-	-	−14.26	1.12E4	−4.49	1.08E6	−7.04	1.83E6	-	2.40E6	n.a.	n.a.
QR	x	x	-	-	x	x	x	-	−126.70	1.14E4	−4.97	1.60E6	−6.90	1.81E6	-	2.41E6	-	**1.18E4**
QR	x	x	-	-	x	x	-	x	−173.18	1.18E4	−6.77	1.35E6	−6.87	1.78E6	-	2.39E6	-	1.34E4
model	**input features**								**AR**		**Helpdesk**		**BPIC-17**		**BPIC-19**		**PCR**	
	a	**r**	**ac**	**rc**	**s**	**d**	**ii1**	**ii3**	**LL**	**LL**	**CRPS**	**CRPS**	**LL**	**CRPS**	**LL**	**CRPS**	**LL**	**CRPS**
LSTM	x	x	(x)	(x)	x	x	-	-	-	1.43E4	−4.15	7.58E5	−7.00	1.56E6	**−13.26**	2.75E6	−4.29	2.09E4
model	**input features**								**AR**		**Helpdesk**		**BPIC-17**		**BPIC-19**		**PCR**	
	a	**r**	**ac**	**rc**	**s**	**d**	**ii1**	**ii3**	**LL**	**CRPS**	**LL**	**CRPS**	**LL**	**CRPS**	**LL**	**CRPS**	**LL**	**CRPS**
opt.	x	x	-	x	x	x	(x)	(x)	*0.55*	*7.96E3*	-	-	-	-	-	-	-	-

fact, learn these relationships, it does not take that advantage to the same extent on the other event logs. However, the quantile regression approach also fails to achieve better results when more features are used. This may be because these relationships are not pronounced in many real-world event logs, or because the tree-based models were unable to capture them with the selected training parameters.

Related work

A precise representation of activity processing and waiting times is a core requirement for a broad range of methods in business process management and has been addressed in approaches for business process simulation (Section "Business process simulation"), predictive process monitoring (Section "Predictive process monitoring"), and resource allocation and scheduling (Section "Resource allocation and scheduling").

Business process simulation

Business process simulation is considered one of the "most established analysis techniques" (van der Aalst 2015) in Business Process Management. Since manually creating simulation models can be cumbersome, data-driven business process simulation approaches have been introduced, which use historical process data to derive a business process simulation model. Many data-driven business process simulation approaches build on the foundational work of Rozinat et al. (2009) and train independent models for the individual process perspectives. Many approaches discover the control-flow perspective, decision points, roles, processing and waiting times, etc., separately and integrate them into one simulation model. In these approaches, processing and waiting times are either obtained from probability distributions that were fit on the event log data, or from (point-estimation) regression models. Clearly, probability distributions can provide a higher variability in their simulation outcomes, but regression models have been used when particular determinants, e.g., the workload, have been identified as decisive (Martin et al. 2016).

In the work of Rozinat et al. (2009), two normal distributions are fit for each activity, one for its waiting time and one for its processing time. In the works of Camargo et al. (2020), López-Pintado et al (2024) the PIX framework is used to fit a probability distribution. In this framework, it is assumed that the processing and waiting times follow a parametric probability distribution. Therefore, several parametric probability distributions, such as (log)normal, exponential, gamma, or uniform distributions, are fit on the historical data, and the one that achieves the highest likelihood on the data is selected afterwards. López-Pintado et al. (2024) further assume that the resource that conducts an activity substantially determines the processing time of the activity. Therefore, they present a *resource differentiated* approach in which they use the PIX-framework to fit probability distributions for the pairs of activities and resources and show that this approach can achieve a higher accuracy of business process simulation models. Some recent approaches have proposed to combine point-estimation regression techniques with probability distributions: Meneghello et al. (2024) propose a business process simulation model in which processing and waiting times can either be derived from probability distributions or obtained from regression models. Similarly, in the business process simulation model from Camargo et al. (2022), two (point-estimation) LSTM models are used. One is for obtaining a processing time, and the other is for obtaining a waiting

time. In the simulator used by Kunkler and Rinderle-Ma (2024), mean regression via a neural network is first employed to predict the expected processing times of an activity. Then, a normally distributed error term is added to the prediction to account for variability.

Predictive process monitoring

Predictive Process Monitoring (PPM) is another field in which temporal predictions of business processes has been addressed. Many approaches have only employed point-estimation prediction models instead of probabilistic models; however, some have addressed similar challenges with encoding contextual information into machine learning models (c.f. Di Francescomarino et al. (2018), Di Francescomarino and Ghidini (2022)). Verenich et al. (2019) conducted a survey on encoding approaches and performed an extensive benchmark for remaining process cycle time predictions using the tree-based ensemble learner XGBoost and compared it to an LSTM model. They could show that for a mean regression, an LSTM model outperforms a tree-based learner on nearly all datasets and feature encoding techniques used for the tree-based learners. Their results for the mean regression case are in contrast to our results, in which the LSTM approach could not outperform a DR-BART model. Recent remaining process cycle time prediction approaches have also employed probabilistic neural network architectures for interval predictions (Weytjens and Weerdt 2022).

Such methods can be regarded as black-box approaches because they do not reveal how their predictions are composed, for example, how individual activities contribute to the eventual cycle time. Some other approaches have therefore proposed white-box approaches in which information about the individual processing and waiting times of activities is aggregated for predicting the remaining cycle time of a business process: Verenich et al. (2019) train a point-estimation regression model for the duration of every activity and obtain a prediction for the remaining process cycle time by aggregating the processing and waiting time predictions for every expected remaining activity of a case. In Rogge-Solti and Weske (2013, 2015), arbitrary probability distributions of processing times of every activity are used for predicting the remaining time of business processes. The approach leverages information from probability distributions to dynamically update the expected finishing time of an activity during its execution. However, their approaches do not address the learning of dynamic probability distributions.

Furthermore, some PPM approaches have been presented that predict the sequence of events until process completion, known as suffix prediction (Tax et al. 2017). Some of these approaches do only predict the event labels but also the corresponding time stamp (Gunnarsson et al. 2023; Taymouri et al. 2021; Wuyts et al. 2024). Kunkler et al. (2025) employ MC sampling on an uncertainty-aware neural network architecture to sample different suffix scenarios which can be used, e.g., for predicting remaining process cycle time intervals.

Resource allocation and scheduling

Information about processing and waiting times of activities is also required in some approaches that address resource allocation or scheduling of business processes. Many approaches have resorted to using only deterministic information about processing and waiting times, such as the mean duration (e.g., in Havur et al. (2016), Kunkler and

Rinderle-Ma (2024), Park and Song (2023)). Probability distributions might not have been employed in these approaches because, although many scheduling problems with deterministic processing times are already computationally hard, incorporating stochastic processing times can make the problem even more challenging (see Rostami et al. (2018)).

Discussion & conclusion

In this work, we presented *Pro3Log*, which enables probabilistic learning of processing and waiting times of activities based on process event logs. To achieve probabilistic learning, *Pro3Log* first extracts durations from event logs and encodes the history of process cases and contextual information into a fixed-sized feature vector. We presented two different *Pro3Log* variants for learning on the encoded data, namely the basic variant in which the entire event log is encoded and subsequently a single DR-BART model trained on the encoded data, and the MoE approach, in which an event log is split into several clusters on which individual DR-BART models are trained.

We evaluated both *Pro3Log* versions in business process simulation on four real-world and one synthetic data sets and compared them with three baseline approaches.

The results on the synthetic data set demonstrate that *Pro3Log* can learn complex relationships for processing and waiting times from event logs. In most settings, the MoE version of *Pro3Log* achieved better results than its basic version and demonstrates the ability to outperform baseline approaches that cannot learn non-parametric probability distributions.

Overall, the results demonstrate that *Pro3Log* can be utilized to create more precise data-driven business process simulation models compared to traditional approaches, which model processing and waiting times either as static probability distributions or only as parametric distributions.

The results also demonstrate that selecting irrelevant feature attributes can decrease the performance of *Pro3Log*, which may be partly explained by the fact that DR-BART models require many more MCMC iterations when irrelevant features are present. Since training for a sufficiently large number of MCMC iterations can become computationally infeasible in practice, future work on learning processing and waiting times from event logs should consider alternative probabilistic models to DR-BART.

Author contributions
MK developed the conceptual approach and the prototypical implementation. Moreover, MK conducted the evaluation and wrote the initial version of the manuscript. MK and SRM discussed the conceptual contribution. SRM reviewed and reworked the initial manuscript.

Funding
Open Access funding enabled and organized by Projekt DEAL.

Data availability
Data is provided within the manuscript: Artificial event log: https://github.com/ltsstar/TaskExecutionTimeMining/blob/main/data/artificial_event_log_2.xes. PCR event log: https://doi.org/10.5281/zenodo.11617408. BPIC-17 event log: https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b. BPIC-19 event log: https://doi.org/10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1. Helpdesk event log: https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb. Code for Pro3Log: https://github.com/ltsstar/TaskExecutionTimeMining/.

Declarations

Ethical approval
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 15 August 2025 / Accepted: 19 December 2025
Published online: 06 February 2026

References

Ali MA, Milani F, Dumas M (2025) Data-driven identification and analysis of waiting times in business processes. Bus Inf Syst Eng 67(2):191–208. https://doi.org/10.1007/S12599-024-00868-5
Camargo M, Dumas M, González O (2020) Automated discovery of business process simulation models from event logs. Decis Support Syst 134:113284. https://doi.org/10.1016/J.DSS.2020.113284
Camargo M, Dumas M, Rojas OG (2022) Learning accurate business process simulation models from event logs via automated process discovery and deep learning. Adv Inf Syst Eng 55–71
Chapela-Campa D, Benchekroun I, Baron O, Dumas M, Krass D, Senderovich A (2025) A framework for measuring the quality of business process simulation models. Inf Syst 127(102447). https://doi.org/10.1016/j.is.2024.102447
Chapela-Campa D, López-Pintado O, Suvorau I, Dumas M (2025) SIMOD: automated discovery of business process simulation models. SoftwareX 30(102157). https://doi.org/10.1016/J.SOFTX.2025.102157
Chipman H, George E, McCulloch R (1998) Bayesian cart model search. J Am Stat Assoc 93(443):935–948. https://doi.org/10.1080/01621459.1998.10473750
Chipman H, George E, McCulloch R (2010) BART: Bayesian additive regression trees. Ann Appl Stat 4(1):266–298. https://doi.org/10.1214/09-AOAS285
Destercke S, Dubois D, Chojnacki E (2008) Unifying practical uncertainty representations–i: Generalized p-boxes. Int J Approx Reason 49(3):649–663. https://doi.org/10.1016/j.ijar.2008.07.003
Di Francescomarino C, Ghidini C (2022) Predictive process monitoring. Process Min Handb 320–346
Di Francescomarino C, Ghidini C, Maggi FM, Milani F (2018) Predictive process monitoring methods: which one suits me best? Bus Process Manag 462–479
Dumas M, Rosa ML, Mendling J, Reijers HA (2018) Fundamentals of business process management, 2nd edn. Springer
Fracca C, de Leoni M, Asnicar F, Turco A (2022) Estimating activity start timestamps in the presence of waiting times via process simulation. Adv Inf Syst Eng 287–303
Gneiting T, Raftery AE (2007) Strictly proper scoring rules, prediction, and estimation. J Am Stat Assoc 102(477):359–378. https://doi.org/10.1198/016214506000001437
Gunnarsson BR, Vanden Broucke S, Weerdt JD (2023) A direct data aware LSTM neural network architecture for complete remaining trace and runtime prediction. IEEE Trans Serv Comput 16(4):2330–2342. https://doi.org/10.1109/TSC.2023.3245726
Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182
Havur G, Cabanillas C, Mendling J, Polleres A (2016) Resource allocation with dependencies in business process management systems. Bus Process Manag Forum 3–19
Hersbach H (2000) Decomposition of the continuous ranked probability score for ensemble prediction systems. Weather Forecast 15(5):559–570. https://doi.org/10.1175/1520-0434(2000)015%3C0559:DOTCRP%3C2.0.CO;2
Hüllermeier E, Waegeman W (2021) Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. Mach Learn 110(3):457–506. https://doi.org/10.1007/S10994-021-05946-3
IEEE standard for extensible event stream (xes) for achieving interoperability in event logs and event streams (2023) IEEE Std 1849–2023 (Revision of IEEE Std 1849-2016), p 1–55. https://doi.org/10.1109/IEEESTD.2023.10267858
Klein N (2024) Distributional regression for data analysis. Annu Rev Stat Appl 11:321–346. https://doi.org/10.1146/annurev-statistics-040722-053607
Kneib T, Silbersdorff A, Säfken B (2023) Rage against the mean – a review of distributional regression approaches. Econom Stat 26:99–123. https://doi.org/10.1016/j.ecosta.2021.07.006
Koenker R, Bassett G Jr. (1978) Regression quantiles. Econometrica 46(1):33–50
Kunkler M, Mustroph H, Rinderle-Ma S (2025) Probabilistic suffix prediction of business processes. Process Min 1–8
Kunkler M, Rinderle-Ma S (2025) Probabilistic learning of temporal uncertainties in business processes. In: Enterprise, business-process and information systems modeling, p 193–208
Kunkler M, Rinderle-Ma S (2024) Online resource allocation to process tasks under uncertain resource availabilities. Process Min 137–144
López-Pintado O, Dumas M, Berx J (2024) Discovery, simulation, and optimization of business processes with differentiated resources. Inf Syst 120:102289. https://doi.org/10.1016/j.is.2023.102289
Mangler J, Rinderle-Ma S (2022) Cloud process execution engine: architecture and interfaces. CoRR, abs/2208.12214, https://doi.org/10.48550/ARXIV.2208.12214
Martin N, Depaire B, Caris A (2016) The use of process mining in business process simulation model construction - structuring the field. Bus Inf Syst Eng 58(1):73–87. https://doi.org/10.1007/S12599-015-0410-4
Mendling J, Leopold H, Meyerhenke H, Depaire B (2025) Methodology of algorithm engineering. ACM Comput Surv 58(4). https://doi.org/10.1145/3769071
Meneghello F, Middelhuis J, Genga L, Bukhsh Z, Ronzani M, Francescomarino CD, Dijkman RM (2024) Optimizing resource allocation policies in realworld business processes using hybrid process simulation and deep reinforcement learning. Business process management, p 167–184
Orlandi V, Murray J, Linero A, Volfovsky A (2021) Density regression with bayesian additive regression trees. arXiv. (arXiv:2112.12259 [stat])
Park G, Song M (2023) Optimizing resource allocation based on predictive process monitoring. IEEE Access 11:38309–38323. https://doi.org/10.1109/ACCESS.2023.3267538
Pufahl L, Stiehle F, Ihde S, Weske M, Weber I (2025) Resource allocation in business process executions - a systematic literature study. Inf Syst 132(102541). https://doi.org/10.1016/J.IS.2025.102541
Pyzdek T, Keller P (2024) Six sigma handbook: a complete guide for green belts, black belts, and managers at all levels, 6th edn. McGraw-Hill Education, New York

Rogge-Solti A, Weske M (2013) Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. Serv Oriented Computing 389–403
Rogge-Solti A, Weske M (2015) Prediction of business process durations using non-markovian stochastic petri nets. Inf Syst 54:1–14. https://doi.org/10.1016/J.IS.2015.04.004
Rosemann M, Recker J, Flender C (2008) Contextualisation of business processes. Int J Bus Process Integr Manag 3(1):47–60. https://doi.org/10.1504/IJBPIM.2008.019347
Rostami S, Creemers S, Leus R (2018) New strategies for stochastic resourceconstrained project scheduling. J Sched 21(3):349–365. https://doi.org/10.1007/S10951-016-0505-X
Rozinat A, Mans R, Song M, van der Aalst W (2009) Discovering simulation models. Inf Syst 34(3):305–327. https://doi.org/10.1016/j.is.2008.09.002
Schonenberg H, Mans R, Russell N, Mulyar N, van der Aalst WMP (2008) Process flexibility: a survey of contemporary approaches. In: Advances in enterprise engineering I, workshop ciao! and workshop eomas, p 16–30
Senderovich A, Francescomarino CD, Ghidini C, Jorbina K, Maggi FM (2017) Intra and inter-case features in predictive process monitoring: a tale of two dimensions. Bus Process Manag 306–323
Senderovich A, Francescomarino CD, Maggi FM (2019) From knowledge-driven to data-driven inter-case feature encoding in predictive process monitoring. Inf Syst 84:255–264. https://doi.org/10.1016/J.IS.2019.01.007
Senderovich A, Weidlich M, Gal A, Mandelbaum A (2014) Queue mining - predicting delays in service processes. Adv Inf Syst Eng 42–57
Senderovich A, Weidlich M, Gal A, Mandelbaum A (2015) Queue mining for delay prediction in multi-class service processes. Inf Syst 53:278–295. https://doi.org/10.1016/J.IS.2015.03.010
Shannon CE (1948) A mathematical theory of communication. Bell Syst Tech J 27(3):379–423. https://doi.org/10.1002/J.1538-7305.1948.TB01338.X
Silverman P (1971) The ten year rule. R U Serv Institution 116(661):42–45. https://doi.org/10.1080/03071847109425987
Slonim N, Friedman N, Tishby N (2001) Agglomerative multivariate information bottleneck. In: Advances in neural information processing systems. MIT Press, p 929–936
Slonim N, Tishby N (1999) Agglomerative information bottleneck. In: Advances in neural information processing systems. The MIT Press, p 617–623
Stertz F, Mangler J, Rinderle-Ma S (2020) Temporal conformance checking at runtime based on time-infused process models. CoRr, abs/2008.07262
Tax N, Verenich I, Rosa ML, Dumas M (2017) Predictive business process monitoring with LSTM neural networks. Adv Inf Syst Eng 477–492
Taymouri F, Rosa ML, Erfani SM (2021) A deep adversarial model for suffix and remaining time prediction of event sequences. In: SIAM international conference on data mining, p522–530
Tishby N, Pereira FCN, Bialek W (2000) The information bottleneck method. CoRR, physics/0004057
van der Aalst WMP (2010) Business process simulation revisited. In: Enterprise and organizational modeling and simulation - workshop, EOMAS, p 1–14
van der Aalst WMP (2015) Business process simulation survival guide. In: Handbook on business process management 1: introduction, methods, and information systems. Springer, Berlin Heidelberg, p 337–370
van der Aalst WMP, Rosa ML, Santoro FM (2016) Business process management - don't forget to improve the process! Bus Inf Syst Eng 58(1):1–6. https://doi.org/10.1007/S12599-015-0409-X
Van Looy A, Shafagatova A (2016) Business process performance measurement: a structured literature review of indicators, measures and metrics. SpringerPlus 5(1):1797. https://doi.org/10.1186/s40064-016-3498-1
Verenich I, Dumas M, Rosa ML, Maggi FM, Teinemaa I (2019) Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. ACM Trans Intell Syst Technol 10(4):34:1–34:34. https://doi.org/10.1145/3331449
Verenich I, Dumas M, Rosa ML, Nguyen H (2019) Predicting process performance: a white-box approach based on process models. J Softw Evol Process 31(6). https://doi.org/10.1002/SMR.2170
Weinzierl S, Zilker S, Dunzer S, Matzner M (2024) Machine learning in business process management: a systematic literature review. Expert Syst Appl 253:124181. https://doi.org/10.1016/J.ESWA.2024.124181
Weytjens H, Weerdt JD (2022) Learning uncertainty with artificial neural networks for predictive process monitoring. Appl Soft Comput 125(109134). https://doi.org/10.1016/J.ASOC.2022.109134
Wombacher A, Iacob M, Haitsma M (2011) Towards a performance estimate in semi-structured processes. In: Service-oriented computing and applications, p 1–5
Wuyts B, Vanden Broucke SKLM, Weerdt JD (2024) Sutran: an encoderdecoder transformer for full-context-aware suffix prediction of business processes. Process Min 17–24
zur Muehlen M, Swenson KD (2010) BPAF: a standard for the interchange of process analytics data. In: Business process management workshops, p 170–181.

Publisher's Note