Prediction of business process durations using non-Markovian stochastic Petri nets

Andreas Rogge-Solti[a,*], Mathias Weske[b]

[a] *Vienna University of Economics and Business, Austria*
[b] *Hasso Plattner Institute, University of Potsdam, Germany*

Abstract

Companies need to efficiently manage their business processes to deliver products and services in time. Therefore, they monitor the progress of individual cases to be able to timely detect undesired deviations and to react accordingly. For example, companies can decide to speed up process execution by raising alerts or by using additional resources, which increases the chance that a certain deadline or service level agreement can be met. Central to such process control is accurate prediction of the remaining time of a case and the estimation of the risk of missing a deadline.

To achieve this goal, we use a specific kind of stochastic Petri nets that can capture arbitrary duration distributions. Thereby, we are able to achieve higher prediction accuracy than related approaches. Further, we evaluate the approach in comparison to state of the art approaches and show the potential of exploiting a so far untapped source of information: the *elapsed time* since the last observed event. Real-world case studies in the financial and logistics domain serve to illustrate and evaluate the approach presented.

Keywords: business processes, duration prediction, risk control, stochastic Petri nets

*Corresponding author
Email addresses: andreas.rogge-solti@wu.ac.at (Andreas Rogge-Solti), mathias.weske@hpi.de (Mathias Weske)

1. Introduction

Managing a company's business processes is critical for its success in a competitive market environment [1; 2]. Not only is it important to strategically align the business processes with the company's goals, the daily operational support of business processes is also necessary to ensure smooth operation [3].

Psychological studies show that waiting time has a negative impact on customer satisfaction [4]. While companies cannot completely avoid waiting times for economic considerations, they can and should use all available information to provide their customers with reliable estimates of remaining time. An approach is to offer waiting time guarantees to customers. Kumar et al. show that these guarantees improve customer satisfaction, if they are met [5]. Such waiting time guarantees are common in various domains (e.g., fast food, travel, finance), where customers get compensated if their waiting time exceeds the agreed thresholds.

The increasing degree of automation by process oriented information systems and the adoption of sensing devices in business processes [6] produce large amounts of process execution data. These data hold valuable information which can be used for analyses in the context of business process intelligence. This paper focuses on the temporal performance aspect of business processes. In fact, we use expressive probabilistic models that we enrich with information extracted from event data [7]. Most prominently, performance models are used to predict remaining durations [8; 9; 10] and to estimate the risk of missing a given deadline [11; 12]

Based on these insights, we offer means to accurately predict remaining durations, and to compute risks to breach temporal deadlines or guarantees. Thereby, we exploit the information of elapsed time since the last event to increase accuracy. These means can be also used to compute reasonable waiting time guarantees, e.g., the waiting time that is met in 99 percent of the cases. Moreover, the approach presented in this paper can be applied in resource management, where an accurate predictor and knowledge about the uncertainty for remaining time of activities is critical for effective scheduling [13].

Summarizing the contributions of this paper, we rely on an expressive stochastic performance model of a business process to:

- predict the expected remaining duration of a business process,

- predict the risk of breaching a temporal deadline,
- use elapsed time as a means to improve prediction accuracy,
- evaluate the model against state of the art approaches.

The remainder of the paper is structured as follows. In Section 2, we formally introduce the concepts and the model used in the prediction approach with a motivating example. Next, in Section 3, we present the approach to predict remaining time including associated confidence intervals, and also predict risk of breaching a temporal boundary. We overview related approaches in Section 4 and highlight differences to and synergies with our approach. Subsequently, Section 5 provides an evaluation with the state of the art and discusses the results. We conclude the paper in Section 6 with reflections on limitations and next steps.

2. Preliminaries and Motivating Example

To best illustrate the approach for prediction, we introduce a running example from the finance domain. Figure 1 depicts a loan approval process as a BPMN [14] diagram. In this process, there are two communicating partners: the client and the financial department of a bank. The client can start a loan process by submitting a loan request to the financial department. After that, the client waits for a reply. Meanwhile, the request is evaluated in the financial department and, depending on the evaluation, the loan is either accepted or declined. If declined, the client is notified and the process terminates. If the loan request is accepted, it first must be finalized and then can be approved by sending a notification.

In this process, the financial department has implemented a service level agreement (SLA) that states that the response will be sent to the client latest after 5 business days. In case of a later response the client is offered a compensation. Also, the management would like to have an estimate of the remaining case durations without having to ask the process participants to avoid excess communication. The vision is to also make the remaining duration estimates available to the affected clients of the bank, to further increase customer satisfaction. In this context, we investigate two issues: Predicting the remaining duration, and estimating the risk of deadline transgression in business processes.

Next, we define event logs that reflect the data gathered from information systems or sensors in a process execution environment. Let A be a set of

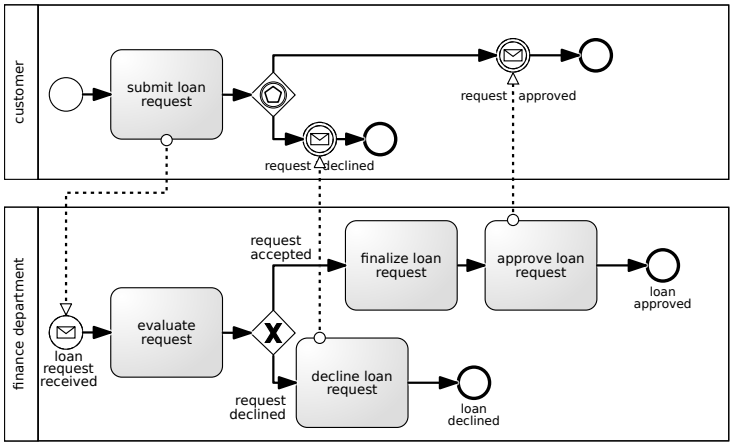


Figure 1: Loan application process as BPMN model. After a client requests a loan by submitting a request to the finance department, it is first evaluated. Depending on the evaluation, the loan can either be declined, or accepted. If accepted, the loan request will be finalized and an approved. The client is notified in any case about the decision.

process activities and TD be the time domain.

Definition 1 (Event Log). *An event log over a set of activities A and time domain TD is defined as $L_{A,TD} = (E, C, \alpha, \beta, \gamma)$, where:*

- *E is a finite set of events*
- *C is a finite set of cases (process instances),*
- *$\alpha : E \to A$ is a function assigning each event to an activity.*
- *$\beta : E \to C$ is a surjective function assigning each event to a case.*
- *$\gamma : E \to TD$ is a function assigning each event to a timestamp. Events are ordered by their timestamps.*

We call the events belonging to the same case a *trace*. That is, the trace of case c is $\{e \in E \mid \beta(e) = c\}$.

Beside the data, we need a model that probabilistically captures durations. The latter will be used to predict remaining durations and the risk of deadline transgression. There exist many modeling notations (e.g., Business Process Model and Notation [14]) that imperatively describe the permissible behavior of business processes. We rely on the Petri net [15] representation of these models, because Petri nets provide a formal model with a mathematical flavor and are able to capture most common control flow patterns [16]. That is, they are able to capture sequential dependencies, exclusiveness, and

concurrency. Moreover, there exist translations from different modeling languages to Petri nets [17], which allow us to effectively apply our prediction approach in various settings. Hence, we can apply the approach independent of the modeling language that is used to capture the business processes by a prior translation of that model into a Petri net.

We first revisit the basic Petri net model, as introduced by Petri in his seminal thesis [15].

Definition 2 (Petri Net). *A Petri net is a tuple* $PN = (P, Tr, F, M_0)$ *where:*

- P *is a set of places.*
- Tr *is a set of transitions.*
- $F \subseteq (P \times Tr) \cup (Tr \times P)$ *is the flow relation.*
- $M_0 \in P \to \mathbb{N}_0$ *is an initial marking.*

Petri nets encode structural relationships between transitions, e.g., the firing of a transition creates tokens on its output places, which enable further transitions. To capture workflow systems, a desirable property of Petri nets was proposed by van der Aalst [16]. Therefore, he introduced the notion of workflow nets. A workflow net has two special places: a *source place*, and a *sink place*. The definition requires that all transitions and places of the net lie on a path between these two places. A token on the source place signifies the start of a case, and when a token reaches the sink place the case is completed. A workflow net is called *sound* [18], if any case will eventually terminate and the moment it reaches completion (with a token on the sink place) all other places are empty. Additionally, there are no dead transitions, i.e., for each transition, there is a firing sequence that includes it. In this paper, we assume that the process models are sound workflow nets.

Besides these properties, we need to capture performance characteristics of individual cases. For that purpose, we enrich transitions with probabilistic delays to reflect activity durations and waiting times. Moreover, we assign weights to transitions to capture the probability of decision outcomes. For example, to accurately predict the remaining duration, it helps to know that a path that usually takes very long, is only taken by 10 percent of the cases, while the other 90 percent follow the regular faster process path. The enriched model for our purposes is called generally distributed transition stochastic Petri net (GDT_SPN) [7], which we define as follows.

Definition 3 (Generally Distributed Transition Stochastic Petri Net). *A* GDT_SPN *is a seven-tuple:* GDT_SPN $= (P, Tr, \mathcal{P}, \mathcal{W}, F, M_0, \mathcal{D})$*, where* (P, Tr, F, M_0) *is the basic underlying Petri net. Additionally:*

- *The set of transitions* Tr *is partitioned into immediate transitions* Tr_i *and timed transitions* Tr_t*.*
- $\mathcal{P} : Tr \rightarrow \mathbb{N}_0$ *is an assignment of priorities to transitions, where* $\forall tr \in Tr_i : \mathcal{P}(tr) \geq 1$ *and* $\forall tr \in Tr_t : \mathcal{P}(tr) = 0$*.*
- $\mathcal{W} : Tr_i \rightarrow \mathbb{R}^+$ *assigns probabilistic weights to the immediate transitions.*
- $\mathcal{D} : Tr_t \rightarrow D$ *is an assignment of arbitrary probability distribution functions* D *to timed transitions, reflecting the durations of the corresponding activities.*

Similar stochastically enriched versions of Petri nets are known in literature, e.g., stochastic Petri net (SPN) [19], generalized stochastic Petri net (GSPN) [20]. The most basic form of probabilistic models, i.e., an SPN [19; 21; 22], is restricted to the use of only exponentially distributed durations. SPN models are efficiently analyzable due to being isomorphic to Markov Chains, which can be efficiently analyzed due to their *memoryless* property. That is, the exponential nature of transition durations abstracts from the fact how long a case has been waiting in a certain state, and there is only a rate of occurrence of the next event, which is constant. GSPN models allow to model immediate transitions with weights, which do not have a stochastic delay associated, but fire immediately. GSPN mdoels have also been shown to be isomorphic to Markov chains [20] and can be analyzed efficiently.

In contrast, we allow arbitrary distributions to better model durations which need at least a minimum amount of time, or to capture deterministic timeouts. GDT_SPN models correspond to what Horváth et al. call non-Markovian stochastic Petri nets [23] in their analysis.

We have shown that we can enrich Petri nets with stochastic execution information gathered from data [24]. In that work an implementation using a maximum likelihood approach is provided open-source in the process mining platform `ProM` [25] in the package `StochasticPetriNets`. The idea is to fit distributions to the observed durations at each transition in the model and learn also the probabilities of decisions in the model.

Now we return to the motivating example. We assume that the financial department has created a business process model for the loan application

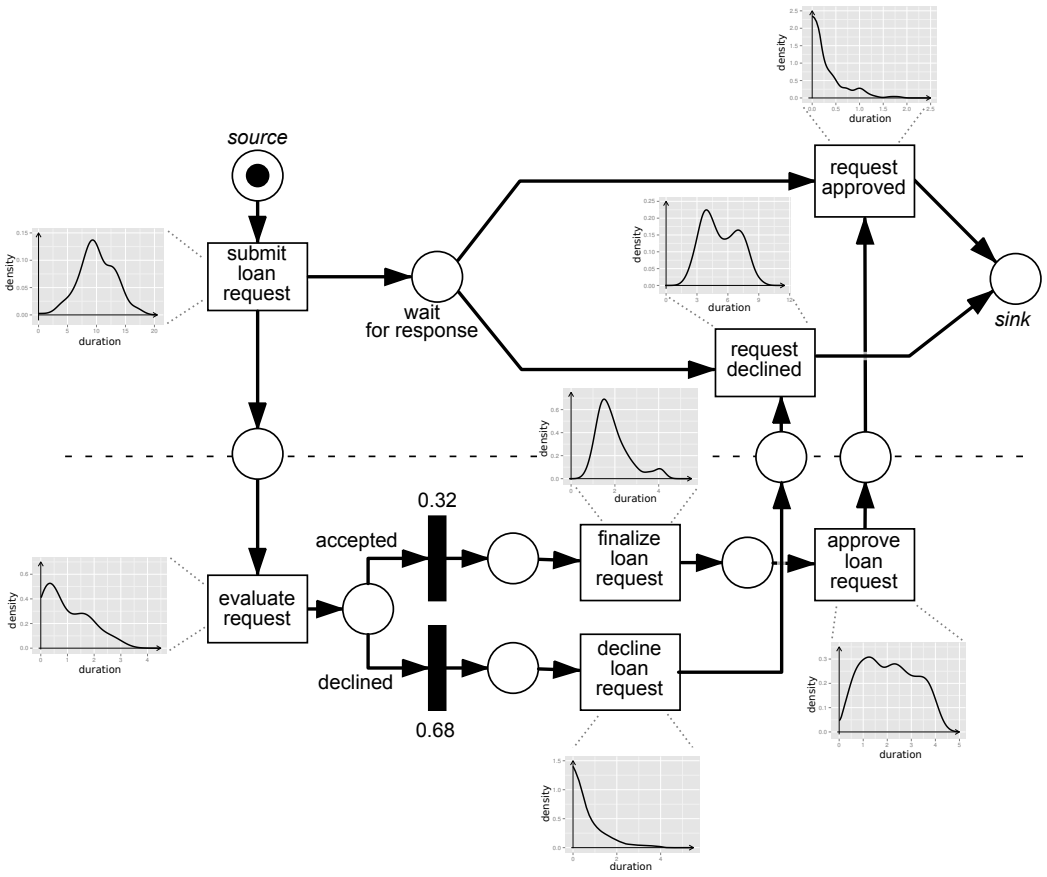


Figure 2: Loan application process as GDT_SPN model. Converted from the BPMN model in Figure 1, cf. [17] and enriched with performance information [24]. Places are circles and transitions are boxes. The model has a *source* and *sink* place indicating start and end. Each timed transition is a white box and is annotated with the corresponding duration distributions. The immediate transitions are filled black bars and are annotated with their weights (e.g., $\mathcal{W}(t_{accept}) = 0.32$) that reflect the probability of firing. That is, loan requests are accepted in 32 percent of the cases, whereas 68 percent get declined. The horizontal dashed line has no semantical meaning, it highlights the boundary of communication between client (above) and financial department (below). The probability density functions associated to timed transitions represent the distribution of firing durations.

process (see Figure 1). In this case, we can either manually, or with the help of automatic translation, convert the BPMN process model to a Petri net model [17]. If we collect and replay historical process execution data that can be extracted from the supporting IT systems, as presented in [24], we obtain a GDT_SPN model. The output model for the running loan application process

example is depicted in Figure 2.

Given a GDT_SPN model that contains decision probabilities and transition durations, and given a running case of the process (i.e., the events that have been produced so far by supporting IT systems), we can compute the remaining case durations and risks of deadline transgressions. In the next section, we shall discuss the core of this contribution: the conceptual advantage of using *elapsed time* since the last event, and the algorithm that allows us to make estimates about the remaining time in GDT_SPN models.

3. Prediction considering elapsed time

In this section, we present the conceptual solution to predicting remaining durations and estimating risk of deadline transgression with a framework based on simulation. The estimation of the remaining time to reach a given state is formalized as follows.

3.1. Estimating the remaining time

Let $t_M \in TD$ be the time when the process reaches the target marking M of interest (e.g., the final marking). We do not know t_M in advance, beacause it is subject to random effects. Hence, we capture it by a random variable X_M. Let t_0 be the current time at estimation, and t_a be the time of finishing the current activity a for a given case. Then, estimating remaining time is to compute the expected value of the remaining relative duration $E(X_M - t_0)$. In [10], we analyzed the accuracy gained by conditioning on the durations of the current activity in the process. Concretely, we compared the two estimates $E(X_M - t_0 \mid t_a \geq t_0)$, and $E(X_M - t_0)$. Note that if there are multiple activities concurrently running, their durations are conditioned accordingly. For example, let t_b be the duration of a second activity b running at time t_0. Then, we compute $E(X_M - t_0 \mid t_a \geq t_0, t_b \geq t_0)$

The resulting estimations can be used in various ways. For example, patients can be informed about their expected length of stay, or the clients awaiting response can be notified with the expected waiting duration.

3.2. Estimating the risk of transgression

This section is about the probability of meeting a certain deadline. Therefore, let t_d be the deadline that should be met. Let again $t_M \in TD$ be the time of reaching the desired marking. Thus, we are interested in estimating the probability $P(t_M > t_d)$ that reflects transgressing the deadline. For this

problem, we also can condition the duration of the current activity on the elapsed time. This results in the formula $P(t_M > t_d \mid t \geq t_0)$, cf. Section 3.1. Note that while the first question yields a relative duration (which is a value greater or equal to 0), the second formula yields a probability between the two extremes of 0 (impossible) and 1 (always).

Based on these estimations, a process controller can find cases in risk of transgressing a deadline, and decide whether to take action to speed up execution. Based on how risk-averse the management is, and the costs inflicted by a transgression of the deadline, the process controller can decide to control the process when a certain risk threshold is exceeded. For example, it can be optimal in terms of cost to only act, when the risk of exceeding a deadline becomes higher than 60 percent.

3.3. Conceptual information gain by capturing elapsed time

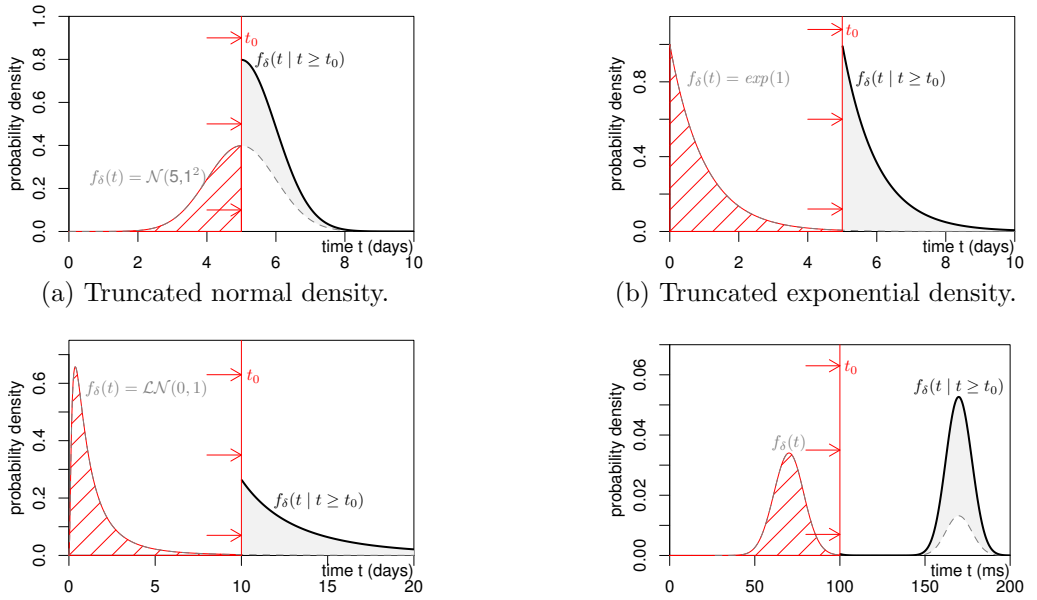
When using a monitoring system that records process steps in an automated way, the events recorded are instantly available to the monitoring system. Our assumption in this work is that the durations of process steps (e.g., human tasks, web service calls) is much greater than the time taken to register the event in the system.

Based on this assumption, a previously neglected source of information can be tapped and exploited to get a more accurate prediction model [10]: Elapsed time since the last event. We illustrate the effects of conditioning on remaining time with characteristic example duration distributions.

Formally, let $t \in TD$ represent time. Let $tr \in Tr_t$ be a timed transition with the assigned duration distribution function $F_\delta = \mathcal{D}(tr)$. We obtain the *density* function f_δ by differentiating the distribution function F_δ, that is, $f_\delta(t) = dF_\delta(t)/dt$. Let $t_0 \in TD, t_0 \geq 0$ be the current time since enabling of tr_i. Let further $f_{\delta_{Dirac}}$ denote the Dirac delta function, which captures the whole probability mass at a single point. We define the density function of the truncated distribution as:

$$f_\delta(t \mid t \geq t_0) = \begin{cases} 0 & t < t_0 \quad F_\delta(t_0) < 1 \\ \frac{f_\delta(t)}{1 - F_\delta(t_0)} & t \geq t_0, \quad F_\delta(t_0) < 1 \\ f_{\delta_{Dirac}}(t - t_0) & \quad F_\delta(t_0) = 1 \end{cases} \tag{1}$$

The part of the density function that is above the threshold t_0 is rescaled such that it integrates to 1, which is a requirement for probability density functions. Note that in the exceptional case that $F_\delta(t_0) = 1$ (i.e., the current



(a) Truncated normal density.

(b) Truncated exponential density.

(c) Truncated lognormal density.

(d) Truncated multi-modal distribution.

Figure 3: Different truncated probability density functions [7]. The current time t_0 truncates the distributions and is depicted as a vertical line.

time t_0 progressed further than the probability density function's support), we use the Dirac delta function with its peak at t_0. In this case, the activity is expected to finish immediately at t_0.

The intuition is as follows. We base our predictions on a stochastic model describing the distribution of a large amount of cases. Thereby, we discard the fraction of the cases that is not consistent with our observation for the current activity's duration, i.e., those cases that would have completed the currently running activity before the current time. In contrast to using conditional probability density functions, traditional methods predict the remaining duration of a case only at event arrival, and subtract elapsed time from the predicted duration at later points in time [26; 8].

Figure 3 shows the effect that *truncation* has on different types of distributions. The density of the normal distribution (depicted in Figure 3a) decreases faster than that of the exponential distribution. Therefore, truncation of the normal distribution makes us more certain that the corresponding event will happen soon. In the figure, the value of the conditional density

function is higher than the original density and more concentrated.

The exponential distribution, depicted in Figure 3b, is a special case, where conditioning on elapsed time does not affect the shape of the distribution, that is, $exp(t) = exp(t + t_0 \mid t > t_0)$. Latter property is an effect of the memoryless property. It implies that the probability that an event will occur in the next minutes is not affected by the time that we spent waiting before. That is, if X is an exponentially distributed random variable, then for any $t, t_0 \geq 0$ it holds that $P(X > t_0 + t \mid t > t_0) = P(X > t)$.

Heavy-tailed distributions (e.g., the lognormal distribution depicted in Figure 3c) are on the other side of the spectrum. Latter distributions can get a higher variance, if we condition on elapsed time. This means that the longer we wait for an event, the less likely it will be that we will observe it in the next minutes. The uncertainty and the expected value of the distribution grows with the time that we spend waiting. In the figure the value of the conditional density is lower and decreasing at a slower rate than the unconditional one.

We can gain the most information about the expected events, when conditioning multi-modal distributions (i.e., distributions with multiple peaks in their density functions). An example is depicted in Figure 3d. The depicted distribution has two modes, that is, its values are clustered at two different time points. Here, once the time progressed beyond the first mode, it can be assumed that the duration will be around 170 milliseconds. This example shows that we can get more accurate results by excluding inconsistent cases from prediction, especially if we use non-parametric methods that are able to approximate distributions with multiple modes.

3.4. Prediction algorithm

Besides the already mentioned assumption of immediate detection of events by the prediction framework, we consider each activity duration in isolation, independently from other activities. This is a common simplifying assumption that we share with all analytical approaches to prediction.

To make predictions for a single case, we require the current state of the case, and the model that captures experiences about the behavior of the process. The prediction algorithm takes five inputs:

1. the GDT_SPN *model* of the business process, cf. Definition 3,
2. the current time t_0,
3. the deadline t_d,
4. the current *trace* of the case c. That is, all observed events up to time t_0. More explicitly, this is $\{e \in E \mid \beta(e) = c \wedge \gamma(e) \leq t_0\}$,

5. the number of simulation iterations n controlling the precision of the prediction.

Algorithm 1 Prediction algorithm

```
1: procedure PREDICT(model, t_0, t_d, trace, n)
2:     currentMarking ← replay(trace, model)          ▷ find current state
3:     times ← new List()                             ▷ used to collect results
4:     for all i ∈ {1, ..., n} do
5:         time ← simulateConstrained(model, currentMarking, t_0)
6:         times.add(time)
7:     end for
8:     return getMean(times), getRatioAbove(times,t_d)
9: end procedure
```

Algorithm 1 describes the procedure. It starts by finding the appropriate current state, i.e., the marking, in the model by replaying the available observed events of the case in the model. We assume that a workflow engine is in charge of controlling the process flow, which facilitates replay of observed events in the log. If this is not the case, we support an alignment technique [27] for replaying non-fitting traces on the model to find the current state.

Next, the algorithm collects simulation results, i.e., completion *times*, of the given number of simulation iterations n in a list. Each simulation run represents a sample from the possible continuations of the process according to the model. Each individual sample is independent and identically distributed, due to the independence assumptions in the model, cf. Definition 3. The `simulateConstrained` method simulates continuations of the trace for the GDT_SPN model, but instead of sampling from the original transition distributions $F_\delta(t)$, it samples from the truncated distributions conditioned on the current time $F_\delta(t \mid t \geq t_0)$, as described in Section 3.3.

The maximum likelihood estimate for the remaining time is the sample mean of the duration samples stored in the list *times* with size n. For convenience, we can access each element in the list with an index $i \leq n$, e.g., $times_3$ is the third duration sample. Then, we approximate $E(t_X - t_0 \mid t \geq t_0)$ by the sample duration values, i.e., we compute:

$$\widehat{E}(t_X - t_0 \mid t_{next} \geq t_0) = \frac{1}{n} \sum_{i=1}^{n} [times_i - t_0] \tag{2}$$

The above measure is a point estimate to the remaining mean duration that does not include our degree of certainty in the measure. We can also compute the confidence interval estimate for the remaining mean duration to be able to say that in most cases (e.g., in 95 percent of the estimations) the interval will contain the real remaining mean duration [28, Chapter 6].

Note that the accuracy of a prediction based on simulated samples depends on both the number of computed samples, as well as the standard deviation within the samples. Therefore, we also support the mode, where instead of a sample size, the user can set required accuracy thresholds. For example, the simulation continues taking samples, until the 99 percent confidence interval on the prediction lies within $\pm$ 3 percent of the predicted value.

For predicting the chance to transgress a target deadline t_d, we use the maximum likelihood predictor. This is simply the ratio of the samples taking more time than the deadline to the number of samples. Let *times* again be our list of samples with size n. Then, we estimate the probability to miss a given deadline t_d as follows:

$$\widehat{P}(t_X > t_d) = \frac{|\{t \in times \mid t > t_d\}| + \epsilon}{n + 2\epsilon} \tag{3}$$

Note that maximum likelihood methods are prone to overfitting the sample, e.g., it is usually not advised to have extreme probabilities of 0 to outcomes that are possible, but were not observed so far. Therefore, we apply *Laplace smoothing* for the estimator with the parameter $\epsilon = 0.5$ which avoids extreme probability values. Chen and Goodman provide an overview of common smoothing techniques [29].

3.5. Open-Source Implementation

We implemented the prediction algorithm in the process mining framework ProM as a plugin[1]. The method to enrich a Petri net with historical performance data extracted from a log is also available in that plugin [24]. It is possible to learn different kind of parametric models (e.g., normally distributed durations), as well as nonparametric models (e.g., simple histograms, or kernel density estimators). If the learned models are only used

[1]Implementation provided open-source in the `StochasticPetriNet` package of ProM. Available at `http://www.promtools.org`

for prediction, simple histograms based on the observed samples suffice for making predictions. In this latter case of histograms, the sampling method can simply pick one of the past observations randomly. We exclude the observations that do not meet the constraint of being greater or equal to the current time t_0. There might be cases, when the current instance takes longer for an activity than all previously observed cases. In this cases, the histogram based sampling returns the constraint t_0 itself.

When expert estimates exist and parametric probability distributions are used in the GDT_SPN model, e.g., normal, exponential, or lognormal distributions, we use rejection sampling. Rejection sampling is simple: we draw a sample from the original distribution and check, whether it meets the given constraints. If not, we reject the sample and repeat the process until we get a sample that meets the constraints.

Before evaluating the prediction method with real data, we introduce related approaches against which we will benchmark the prediction quality.

4. Related Approaches

There are many dimensions that one can predict (e.g., time, costs, resource allocation), however, we focus on the time aspect in this paper. Concretely, we want to predict (i) the remaining time of a current case until it reaches a target state, and (ii) the risk of missing a given temporal deadline for a target state. Therefore, we group related approaches to either of these two categories. First, we discuss approaches predicting remaining time.

Remaining Time Prediction Approaches

In their work, van der Aalst et al. [8] use the available information in logs to predict the remaining case duration based on observed durations in the past. They first create an annotated transition system from the event log. In that state transition system, they collect remaining durations for each visited state from the traces in the log. Finally, when a given case is executed, in each state the remaining time is predicted as the average remaining duration of the historical cases that passed that state. Our approach is similar in the sense that it also abstracts from data and resources, but uses GDT_SPN models instead of transition systems, making our approach more accurate when parallelism exists in the process. The work in [8] serves as one of the benchmarks for the prediction method proposed in this paper.

Building on the work in [8], Folino et al. [9] present an improvement based on predictive clustering. They make use of additional contextual information of a trace (e.g., the current workload in the system) to perform clustering. The idea is to group similar traces and base predictions for new ones with similar features on only similar historical cases of the log. They use the predictions to warn in case of a predicted transgression of a threshold. It seems promising to combine the work in [9] with a GDT_SPN approach.

Other work for prediction of performance was presented by Hwang et al. [30] and similarly Zheng et al. [31]. They use formulae to compute quality of service criteria, such as expected durations of compositions. Typically, these works assume the service compositions to be composed of building blocks, that is, of single-entry single-exit blocks, cf. the formal definition in the work by Kiepuszewski et al. in [32]. The methods proposed can be used for business processes, too. However, the block-structured assumption is lifted in this work, allowing for more complex models, and we target already running instances.

Closely related to our approach is the prediction method presented by Wombacher and Iacob [33]. In their work, the authors prepare event logs of unstructured processes for prediction of activity durations. They use mean duration of activities for predictions, but they do not use runtime information of elapsed time since the last event.

Also based on the assumption that activity durations are normally distributed, the work by Anklesaria et al. estimates completion time for PERT networks [34]. PERT networks are used to model projects with required activities that are in dependency relations, e.g., an activity can only begin after two previous activities have been completed. Optionality and loops are not captured in such models, however, and the remaining project time is determined by the longest path through the network. Anklesaria et al. investigate the effects of correlations between different paths to increase the accuracy of the predicted error by taking into account multiple paths.

Recently, Senderovich et al. proposed to mine queueing networks from logs to enable remaining time prediction [35]. Their work yields high accuracy as resources are captured in their model, but it also requires explicit information about waiting time, i.e., the entry time and exit time of queues at activity stations. A more general approach is followed by de Leoni et al. [36], where a framework is presented that maps the prediction problem in business processes to a well-known data mining setting. That is, they map each event of an event log to a table with several associated attributes to

learn associations and regressors. Thereby, they allow to incorporate further data. However, they do not treat parallelism in their approach. The resulting predictor is similar to the method mentioned in [8], but with more data on which to condition the remaining time.

Simulation has also been proposed and used for operational decision making by Rozinat et al. [37; 38]. The idea is to set up a simulation environment capturing the current situation and start a short-term simulation from this state with different simulation parameters. Their use of simulation is for operational decision support and is focused on the overall performance of business processes. In contrast, we use simulation to make a prediction for the current instance only and use the current elapsed time as additional input to the simulation which allows to improve single predictions.

Analysis of non-Markovian stochastic Petri nets (i.e, GDT_SPN models in this paper) has already been done before. Monte Carlo simulation is the preferred choice for analysis, e.g., in [39; 40]. However, previous work does not address remaining durations of single instances with conditional probability densities.

4.1. Approaches dealing with process related risks

The interest in risk-aware business process management is increasing since one of the earliest works put emphasis on the topic [41]. A recent survey on the topic is provided by Suriadi et al. [42]. While the earlier works approach the problem from a qualitative view, recently quantitative approaches were also proposed. The works of Pika et al. [12], Goluch et al. [43], and Conforti et al. [11] are closely related to our approach.

Pika et al. [12] define indicators for the risk of deadline violations. They search for patterns, such as abnormal activity durations, and use that information for predicting whether a case will be late. Goluch et al. design a prototype for risk-aware simulation with the special focus of simulating threats to resources, e.g., power outage, or illness of a process participant [43]. In contrast, we focus on the risk of missing a temporal deadline. Further, Kang et al. [44] advocate business process monitoring in real time. Their approach is based on classifying historical traces in correct and incorrect traces by data mining techniques, such as support vector machines. Their goal is also to predict and classify current instances, e.g., if they are likely to exceed deadlines. Their approach only captures sequential processes, however, and timestamps of events are not considered in the prediction.

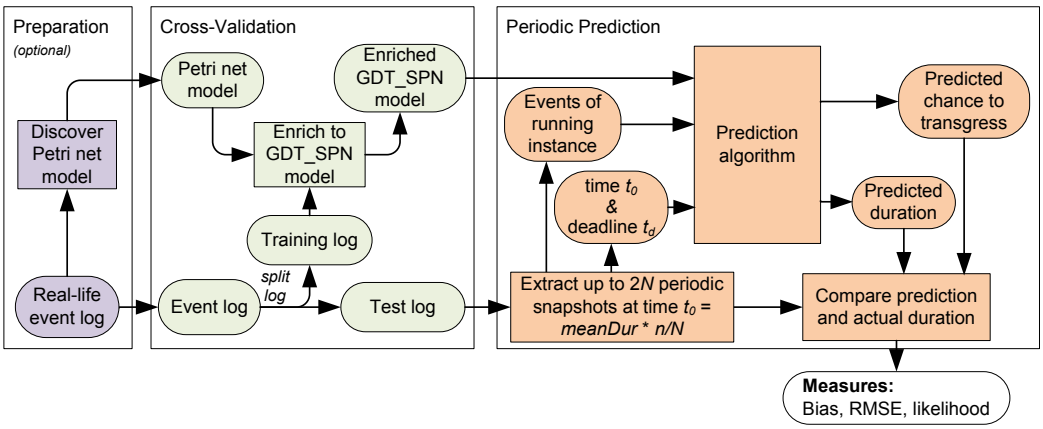


Figure 4: Evaluation setup. The figure contains an optional preparation phase, which we require if we do not have a Petri net model for real-life event logs. After preparation, i.e., simulation of a GDT_SPN model and the extraction of its underlying Petri net, the cross validation is performed. Here, the log is split into training and test log. The training log is used to enrich the GDT_SPN model. The test log is used for evaluating the predictions of the model. We measure prediction accuracy at $2N$ equidistant points in time, where the mean case duration is at the N^{th} measurement.

The work presented by Leitner et al. [45] considers running instances. They use regressions for durations between two-point measures in the process. The predictions are then used to identify whether a service level agreement will be violated. This method is By contrast, our work includes knowledge of the whole business process model to make predictions and we use the elapsed time since the last event as constraining factor.

In summary, the approach presented here is the first one to take elapsed time into account for prediction of remaining durations. In the following, we evaluate the approach with state-of-the art approaches.

5. Evaluation

The evaluation is performed in two parts: a qualitative analysis, and a scalability analysis. In the qualitative analysis, we assess the prediction quality of predicting the remaining durations of a process. We further assess the estimation of the risk of transgressing a given deadline.

5.1. Experiment setup

The procedure works as depicted in Figure 4. Depending on whether we already have a Petri net model for an event log, we need a preparation phase to discover [46; 47] a suitable Petri net model representing the structure of a process.

Given the log and the Petri net input, we perform a 10-fold cross validation with the event log, i.e., we split it into ten parts, use nine parts to enrich the Petri net model to a GDT_SPN model and use the left-out part as test log. The test log represents new cases as they would appear in reality. The experiment is repeated such that each of the ten parts are used once as test log.

For each trace in the test log, we perform $2N$ periodic prediction iterations. Thereby, we run the N^{th} prediction *meanDur* after the start of the case (*meanDur* is the average case duration). In our experiments N is 20, such that we have 40 prediction iterations in total. Each time we perform a prediction, we supply the current time t_0, the target deadline t_d, and the current *trace* up to t_0 to the algorithm. In our approach the model is a GDT_SPN model, but we also compare with the prediction approach proposed in [8]. In latter case, the prediction model is a transition system mined from the event log. The approach in [8] allows different configurations of the transition system. States can be very informative, e.g., encode the entire *sequence* of all events observed so far. But they can also be rather general, e.g., only encoding the single *last* observed event. Therefore, we compare to different configurations of the transition system based approach. As baseline, we also compare to a predictor that predicts the remaining time to be the difference to the mean case duration. In case a prediction would be negative (i.e., when the current time exceeded the mean duration) this predictor returns 0.

5.2. Loan approval process example

Our motivating example of Section 2 is inspired by the process that was presented in the Business Process Intelligence Challenge 2012 [48]. The loan application process of a Dutch financial institute was offered for public analysis. We depict the Petri net representation of the top level process in Figure 5. The process starts with a submission of a loan request. The request can be immediately declined, or be preaccepted by the system. Then, a clerk can accept the loan request to later finalize, and approve it. At these stages, the client also can cancel the loan request, or it can be declined by the clerk.

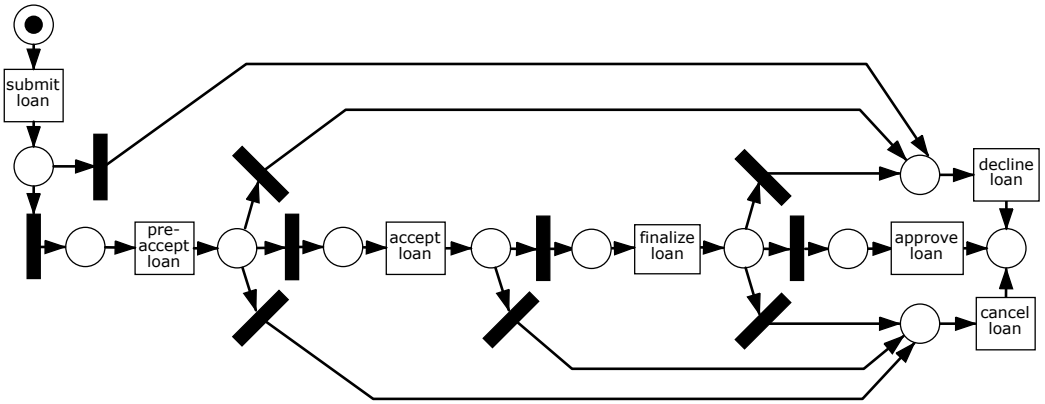


Figure 5: This Petri net process model captures a high level view on a loan application process. Besides the sequential *happy path* (i.e., submitting, preaccepting, accepting, finalizing, and approving the loan), there is also the option for the client to cancel the loan or for the clerk to decline it.

The average duration of all cases in this log is almost nine days, while many cases are quickly declined by the system.

5.3. Shipment import process example

The second real-life process analyzed in our evaluation is from the logistics domain. Figure 6 depicts the model of the shipment import process. Each case reflects the events associated with one container that passes the Dutch harbor to its destination in Europe. The event log contains event entries for arrival by sea, discharge, and subsequent pick up of the containers at the logistics provider. The average duration of the shipment process is four days. Due to the high grade of automation in this process, the fitness between the event log and the process model is perfect.

Giving an accurate estimation of the time of arrival of a container to clients is helpful for their subsequent planning and allows to save warehousing

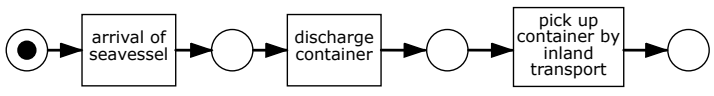


Figure 6: The process model captures three sequential stages of a logistics process: the arrival of sea vessels, container discharges, and further inland shipping.

costs by just in time delivery. It is equally important to have accurate models that can estimate the chance to miss a given shipment guarantee.

Note that the number of activities in this process are few (sparse monitoring points) and therefore, conditioning on each activity's duration should yield higher benefits compared to a process with a large number of activities (dense monitoring points).

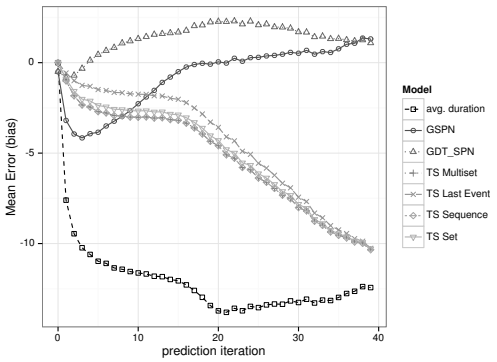
5.4. Experiment results

To assess the proposed approach, we evaluate (i) the precision of the predictions, (ii) the accuracy of the predictions, and (iii) the model fit to predicting missing a given deadline. We measure each of these dimensions for the compared models in each of the 40 periodic predictions as explained in Section 5.1.

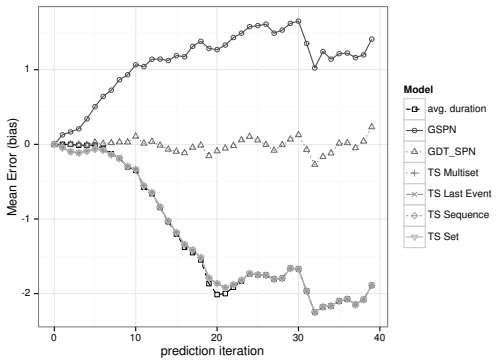
Precision (i) tells us how far the predictors are away from the actual value on average. We measure precision with the model *bias*, for which we use the expected error (i.e., the mean of the prediction errors) as an estimator. In other words, this measure tells us, if our model tends to over- or underestimate the remaining duration. A bias of 0 is optimal. A negative value indicates underestimation, a positive value indicates overestimation.

We measure the accuracy (ii) by the root mean square error (RMSE). The measure is the square root of the average squared errors. It is a common measure that tells us, how close the predicted value is to the real value. The lower this measure is, the more accurate is the prediction model.

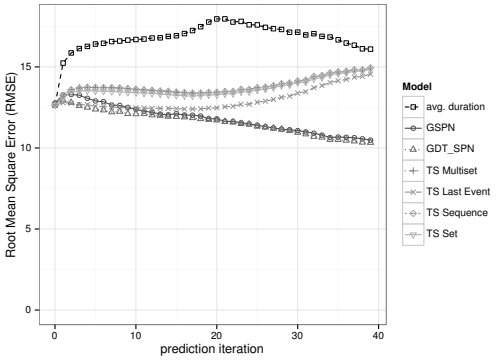
Last, we measure the model's fit (iii) to predicting missing a given deadline t_d by the *log-likelihood* of the model given the data. The deadline t_d is twice the mean case duration for our experiments. In this setting, the data encode whether cases exceed the deadline. Models predict either of these exclusive outcomes with probabilities summing to 1. For example, a model can predict at a given point that the current case will exceed the deadline with a probability of 0.1, and be finished within the deadline with probability 0.9. Let us assume that we observe 2 cases transgressing the deadline and 3 cases meeting the deadline in this case, and that the probabilities remain the same. Then, the log likelihood is simply $\log(0.1^2 \cdot 0.9^3) = -2.14$. A better model in this example would predict that the chance to transgress is 0.4, and the chance to meet a deadline is 0.6. Latter model would yield a higher log likelihood of -1.46. The higher the log-likelihood, the better. Although the measure by itself is not too informative as it also depends on the number of observations, we can estimate a model's relative performance



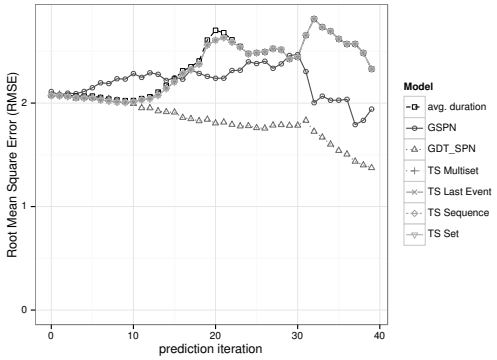
(a) Prediction *bias* in days for the loan approval process [48].



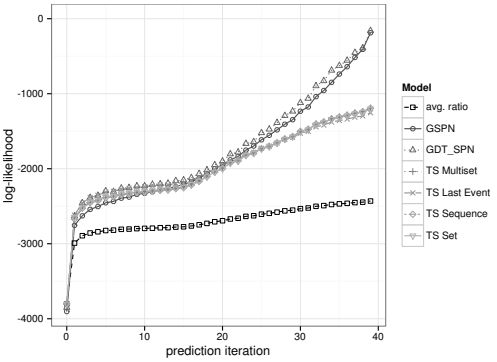
(b) Prediction *bias* in days for the shipment import process.



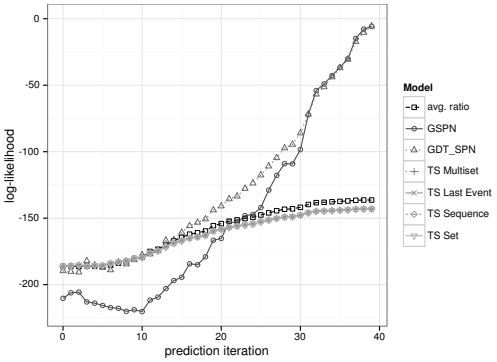
(c) Prediction *accuracy*. RMSE in days for the loan approval process.



(d) Prediction *accuracy*. RMSE in days for the shipment import process.



(e) Model log-likelihoods of deadline transgression for the loan approval process.



(f) Model log-likelihoods of deadline transgression for the shipment import process.

Figure 7: Prediction quality for the logistics process. The root mean square errors are collected for periodic prediction iterations of the case studies. Thereby, a 10-fold cross validation is performed on the data with 40 periodic predictions, s.t. the 20th iteration is at the mean duration. The deadline t_d is set to be twice the mean duration.

by comparing its log-likelihood with another model's log-likelihood in each prediction iteration.

5.5. Discussion of the qualitative analysis

Based on the assumption that the time from occurrence of events to their detection is negligible in comparison to activity duration, we proposed to use the elapsed time as conditioning factor on the duration distributions. We compared various prediction algorithms on two real life event logs.

In Figure 7a, we see that the baseline model underestimates the durations of the loan approval process. This can be explained by many cases automatically rejected at an early stage, but still considered in the remaining predictions. After a first underestimation, the GSPN model assuming exponential distributions performs very well in this case. Our GDT_SPN approach based on truncated duration distributions tends to overestimate the remaining durations, but in later predictions yields comparable results to the GSPN model. The transition system based models start to underestimate the longer running cases around the 15^{th} prediction iteration, as they do not properly condition on the remaining time. These effects are more pronounced in Figure 7b, where our approach is rather unbiased (i.e., the bias is close to 0). The GSPN model is not so suited, however, as the exponential distribution assumption overestimates remaining times in this process. The transition system based approaches cannot exploit the simple structure of the process, and collapse with the single last event predictor.

Regarding prediction *accuracy*, we see that in Figure 7c, the two Petri net based predictors start to outperform the transition system based predictors at later prediction iterations. The GSPN based predictor yields worse results in the shipment import process, cf. Figure 7d. Here, the implicit exponential assumption made by GSPN models does not hold. Note that our approach yields significantly smaller errors than the compared predictors, especially after some time has passed.

We can see in Figures 7e and 7f that the log likelihoods for all models increase over time. This is mostly caused by the decrease in the number of cases still running at later predictions. Nevertheless, the two Petri net approaches prove to have better fitness in providing answers to the question whether a case will take longer than twice the usual time. Especially at later prediction periods it is required to condition on elapsed time to predict whether the current case will take longer than the target deadline.

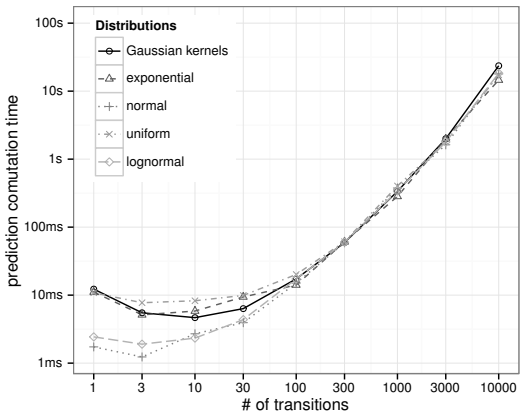


Figure 8: Prediction algorithm durations for differently sized randomly generated, acyclic GDT_SPN models. Axes in log-scale.

To summarize the qualitative evaluation results, the following characteristics of our approach can be observed:

- At early prediction periods, the proposed approach performs about as well as the benchmarks.
- The improvements become more pronounced as time proceeds.
- The approach is most valuable for long running cases, which are critical to be detected and avoided.
- If the duration distributions are not exponential, we only get unbiased predictors, by using the proposed approach with truncated distributions.

Note that we did not compare to prediction approaches clustering cases by system load [9], or using queueing models [35]. This is intentional, as these further conditions are orthogonal to the time domain and could be integrated with the approach presented in this work.

5.6. Scalability Analysis

We propose to use simulation to predict the remaining service execution time for a running case. Therefore, it is interesting to see how long the prediction approach takes to simulate the remaining behavior of the process. To test for scalability, we conduct the following experiment.

First, we randomly generate structured, acyclic GDT_SPN models of different sizes by iterative insertion of sequential, parallel, and exclusive blocks, until we reach the desired node size of the net. For each timed transition

we specify a distribution, i.e., either *uniform*, *normal*, or *lognormal* distributions, or non-parametric *Gaussian kernel* density estimators based on a random number of observations. In general, the run-time of the prediction algorithm depends on multiple factors:

- The *average number of transitions* that need to fire to reach the end of the process, influenced by the size of the net, the progress of the current case, and potential cycles.
- The *kind of transition distributions*, as there exist distributions that are rather costly to draw samples from, e.g., complicated non-parametric models, as well as very simple models, e.g., the uniform distribution.
- The *requested accuracy of the prediction*. Besides the fact that computing a narrow confidence interval takes more samples than allowing more sampling error, the variance of the process durations also influences the number of samples required to achieve the requested precision.
- The *computing power* of the system running the simulation.

For our experiments, we fixed the *requested accuracy* to a 99 percent confidence interval within maximum ±3 percent of error of the mean remaining duration. We control for the variance of the activity durations when assigning the distributions. Regarding *computing power*, we used a desktop computer with a Pentium i5-4570 (3.20 GHz cores) equipped with 16GB of ram. We varied the other two factors, i.e., the *average number of transitions* and the *kinds of distributions* used. Figure 8 depicts the average time taken for remaining time prediction of acyclic GDT_SPN models based on the number of transitions in the model in log-scale. For example, predicting the duration of a model with 300 transitions takes well below 100 milliseconds for all distributions. Note that a prediction of a model with 10000 transitions still is feasible in less than 100 seconds with these configurations. We can see that the computation time scales almost linearly with the size of the model.

In our experience, most business processes involving human activities take hours, days, or sometimes even months to complete. In these situations, the quality of the prediction is more important than the performance of the prediction approach. If performance is critical however, the approach could be extended to provide a fall back option to an analytical method based on GSPN models, as provided by Zimmermann [40].

6. Conclusion

This paper presented an approach for predicting the remaining time, and for estimating the chance of missing a given deadline. The approach is applicable with a limited number of assumptions and yields significant improvements over related work by conditioning on elapsed time. The effects of conditioning on time are most prominent when there are long time spans between observable events in a business process. The underlying model is able to reflect concurrency in processes better than the state of the art [8] and its derivatives, as it builds on the Petri net formalism.

Future work includes taking resource dependencies, or dependencies between activity durations into account. In many cases, process data plays an important role for the remaining durations of activities, and often affects decisions in the process. Adding data dependencies to the model, as for example investigated in [38], is complementary to the proposed approach and promising as well.

References

[1] M. Weske, Business Process Management: Concepts, Languages, Architectures, 2nd Edition, Springer, 2012.

[2] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Springer, 2013.

[3] B. Mutschler, M. Reichert, J. Bumiller, Unleashing the effectiveness of process-oriented information systems: Problem analysis, critical success factors, and implications, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 38 (3) (2008) 280–291.

[4] S. Taylor, Waiting for service: The relationship between delays and evaluations of service, Journal of Marketing 58 (2) (1994) 56–69.

[5] P. Kumar, M. U. Kalwani, M. Dada, The impact of waiting time guarantees on customers' waiting experiences, Marketing Science 16 (4) (1997) 295–314.

[6] S. Meyer, A. Ruppen, C. Magerkurth, Internet of things-aware process modeling: Integrating iot devices as business process resources, in: Advanced Information Systems Engineering, Vol. 7908 of LNCS, Springer, 2013, pp. 84–98.

[7] A. Rogge-Solti, Probabilistic estimation of unobserved process events, Ph.D. thesis, Hasso Plattner Institute at the University of Potsdam, Germany (April 2014).

[8] W. M. P. van der Aalst, M. H. Schonenberg, M. Song, Time prediction based on process mining, Information Systems 36 (2) (2011) 450–475.

[9] F. Folino, M. Guarascio, L. Pontieri, Discovering Context-Aware Models for Predicting Business Process Performances, in: OTM 2012, Springer, 2012, pp. 287–304.

[10] A. Rogge-Solti, M. Weske, Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays, in: ICSOC, Vol. 8274 of LNCS, Springer, Heidelberg, 2013, pp. 389–403.

[11] R. Conforti, M. de Leoni, M. La Rosa, W. M. P. van der Aalst, Supporting risk-informed decisions during business process execution, in: Advanced Information Systems Engineering, Vol. 7908 of LNCS, Springer, 2013, pp. 116–132.

[12] A. Pika, W. M. P. van der Aalst, C. J. Fidge, A. H. M. ter Hofstede, M. T. Wynn, Predicting deadline transgressions using event logs, in: BPM Workshops 2012, Vol. 132 of LNBIP, Springer, 2013, pp. 211–216.

[13] B. Denton, J. Viapiano, A. Vogl, Optimization of surgery sequencing and scheduling decisions under uncertainty, Health Care Management Science 10 (1) (2007) 13–24.

[14] O. M. G. (OMG), Business Process Model and Notation (BPMN) 2.0 Specification (January 2011).

[15] C. A. Petri, Kommunikation mit Automaten, Ph.D. thesis, Technische Hochschule Darmstadt (1962).

[16] W. M. P. van der Aalst, The application of petri nets to workflow management, Journal of Circuits, Systems, and Computers 8 (1) (1998) 21–66.

[17] N. Lohmann, H. M. W. E. Verbeek, R. Dijkman, Petri Net Transformations for Business Processes - A Survey, in: Transactions on Petri Nets and Other Models of Concurrency II, Vol. 5460 of LNCS, Springer, 2009, pp. 46–63.

[18] W. M. P. van der Aalst, Verification of workflow nets, in: ICATPN'97, Vol. 1248 of LNCS, Springer, 1997, pp. 407–426.

[19] F. J. W. Symons, Modeling and Analysis of Communication Protocols Using Numerical Petri Nets, Ph.D. thesis, University of Essex, Great Britain (1978).

[20] M. A. Marsan, G. Conte, G. Balbo, A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems, ACM TOCS 2 (2) (1984) 93–122.

[21] S. Natkin, Reseaux de Petri Stochastiques, Ph.D. thesis, Conservatoire National des Arts et Métiers (CNAM), Paris, France (1980).

[22] M. K. Molloy, Performance Analysis Using Stochastic Petri Nets, Computers, IEEE Transactions on 100 (9) (1982) 913–917.

[23] A. Horváth, A. Puliafito, M. Scarpa, M. Telek, Analysis and Evaluation of Non-Markovian Stochastic Petri Nets, in: Computer Performance Evaluation. Modelling Techniques and Tools, Springer, 2000, pp. 171–187.

[24] A. Rogge-Solti, W. M. P. van der Aalst, M. Weske, Discovering stochastic petri nets with arbitrary delay distributions from event logs, in: BPM Workshops, Vol. 171 of LNBIP, Springer, Heidelberg, 2014, pp. 15–27.

[25] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. Weijters, W. M. P. van der Aalst, The ProM framework: A new era in process mining tool support, in: Applications and Theory of Petri Nets 2005, Springer, 2005, pp. 1105–1116.

[26] B. F. van Dongen, R. A. Crooy, W. M. P. van der Aalst, Cycle time prediction: When will this case finally be finished?, in: OTM 2008, Springer, 2008, pp. 319–336.

[27] W. M. P. van der Aalst, A. Adriansyah, B. F. van Dongen, Replaying history on process models for conformance checking and performance analysis, in: WIREs: Data Mining and Knowledge Discovery, Vol. 2, Wiley Online Library, 2012, pp. 182–192.

[28] W. J. Dixon, F. J. Massey, et al., Introduction to statistical analysis, Vol. 344, McGraw-Hill New York, 1969.

[29] S. F. Chen, J. Goodman, An Empirical Study of Smoothing Techniques for Language Modeling, in: Proceedings of the 34th annual meeting on Association for Computational Linguistics, Association for Computational Linguistics, 1996, pp. 310–318.

[30] S.-Y. Hwang, H. Wang, J. Tang, J. Srivastava, A Probabilistic Approach to Modeling and Estimating the QoS of Web-Services-Based Workflows, Information Sciences 177 (23) (2007) 5484–5503.

[31] H. Zheng, J. Yang, W. Zhao, A. Bouguettaya, QoS Analysis for Web Service Compositions Based on Probabilistic QoS, in: Service-Oriented Computing 2011, Springer, 2011, pp. 47–61.

[32] B. Kiepuszewski, A. H. M. t. Hofstede, C. J. Bussler, On Structured Workflow Modelling, in: Advanced Information Systems Engineering, Vol. 1789 of LNCS, Springer, 2000, pp. 431–445.

[33] A. Wombacher, M.-E. Iacob, Estimating the Processing Time of Process Instances in Semi-structured Processes–A Case Study, in: Services Computing (SCC), 2012 IEEE Ninth International Conference on, IEEE, 2012, pp. 368–375.

[34] K. P. Anklesaria, Z. Drezner, A multivariate approach to estimating the completion time for PERT networks, The Journal of the Operational Research Society 37 (8) (1986) 811–815.

[35] A. Senderovich, M. Weidlich, A. Gal, A. Mandelbaum, Queue mining–predicting delays in service processes, in: CAiSE, Springer, 2014, pp. 42–57.

[36] M. de Leoni, W. M. van der Aalst, M. Dees, A general framework for correlating business process characteristics, in: BPM, Vol. 8659 of LNCS, Springer, 2014, pp. 250–266.

[37] A. Rozinat, M. T. Wynn, W. M. P. van der Aalst, A. H. M. ter Hofstede, C. J. Fidge, Workflow simulation for operational decision support, Data & Knowledge Engineering 68 (9) (2009) 834–850.

[38] A. Rozinat, R. S. Mans, M. Song, W. M. P. van der Aalst, Discovering simulation models, Information Systems 34 (3) (2009) 305–327.

[39] G. Balbo, G. Chiola, Stochastic Petri Net Simulation, in: Proceedings of the 21st conference on Winter simulation, ACM, 1989, pp. 266–276.

[40] A. Zimmermann, Modeling and Evaluation of Stochastic Petri Nets with TimeNET 4.1, in: VALUETOOLS'12, IEEE, 2012, pp. 54–63.

[41] M. Rosemann, M. zur Mühlen, Integrating risks in business process models, in: ACIS 2005 Proceedings, 2005, paper 50.

[42] S. Suriadi, B. Weiss, A. Winkelmann, A. H. ter Hofstede, M. Adams, R. Conforti, C. Fidge, M. L. Rosa, C. Ouyang, M. Rosemann, A. Pika, M. Wynn, Current research in risk-aware business process management: overview, comparison, and gap analysis, Communications of the Association for Information Systems 34 (1) (2014) 933–984.

[43] G. Goluch, A. Ekelhart, S. Fenz, S. Jakoubi, S. Tjoa, T. Mück, Integration of an ontological information security concept in risk aware business process management, in: HICSS-41, 2008, pp. 377–386.

[44] B. Kang, D. Kim, S.-H. Kang, Periodic Performance Prediction for Real-time Business Process Monitoring, Industrial Management & Data Systems 112 (1) (2012) 4–23.

[45] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar, F. Leymann, Runtime Prediction of Service Level Agreement Violations for Composite Services, in: ICSOC/ServiceWave 2009 Workshops, Springer, 2010, pp. 176–186.

[46] W. M. P. van der Aalst, T. Weijters, L. Maruster, Workflow mining: Discovering process models from event logs, Knowledge and Data Engineering, IEEE Transactions on 16 (9) (2004) 1128–1142.

[47] S. J. J. Leemans, D. Fahland, W. M. van der Aalst, Discovering block-structured process models from event logs containing infrequent behaviour, in: BPM Workshops, Vol. 171 of LNBIP, Springer, 2014, pp. 66–78.

[48] B. F. v. Dongen, BPI Challenge 2012 Dataset, `http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f` (2012).