Prediction of Remaining Service Execution Time Using Stochastic Petri Nets with Arbitrary Firing Delays

Andreas Rogge-Solti and Mathias Weske

Business Process Technology Group,
Hasso Plattner Institute, University of Potsdam, Germany
{andreas.rogge-solti,mathias.weske}@hpi.uni-potsdam.de

Abstract. Companies realize their services by business processes to stay competitive in a dynamic market environment. In particular, they track the current state of the process to detect undesired deviations, to provide customers with predicted remaining durations, and to improve the ability to schedule resources accordingly. In this setting, we propose an approach to predict remaining process execution time, taking into account *passed time* since the last observed event.

While existing approaches update predictions only upon event arrival and subtract elapsed time from the latest predictions, our method also considers expected events that have *not* yet occurred, resulting in better prediction quality. Moreover, the prediction approach is based on the Petri net formalism and is able to model concurrency appropriately. We present the algorithm and its implementation in ProM and compare its predictive performance to state-of-the-art approaches in simulated experiments and in an industry case study.

Keywords: business process performance, remaining time prediction, stochastic Petri nets, generally distributed durations, conditional probability.

1 Introduction

Organizations provide services to their customers. To provide a high degree of flexibility, these services are implemented by business processes. As a result, business process management (BPM) plays an important role to improve the performance and quality of business processes [22]. Customers and clients compare the range of goods and services of competitors in a world-wide market, increasing the pressure on companies to conduct their business efficiently and effectively. In particular, a company needs to monitor and control its processes in case of delays, or other unexpected events, to meet given service level agreements.

Predicting the remaining process duration and detecting and preventing risks has spawned much interest recently [13,4,8,18]. The motivation is to ensure customer satisfaction by increasing the overall ratio of services that complete within defined thresholds. Accurate prediction methods are essential to prevent exceptionally long service times. Predictions of remaining service times of a running case can be used in other scenarios, as well. One scenario is to provide customers with information about the progress and expected duration of their case. Further, the predicted remaining duration can be used internally, e.g., for resource scheduling.

S. Basu et al. (Eds.): ICSOC 2013, LNCS 8274, pp. 389–403, 2013.

Several models exist for prediction purposes. Often, the simplifying assumption of exponentially distributed durations is made [17], because the resulting models can be analyzed efficiently. However, if we also want to capture deterministic time-outs or irregularities like multiple modes of distributions, we need more expressive performance models. Current prediction algorithms proposed in [4,8] only update the predictions upon arrival of events, or are based on exponential distributions [25]. In contrast to this, we take advantage of passed time as an influencing and restricting factor of predictions. In other words, we make use of knowledge about expected events *not observed yet*, and thereby significantly improve the prediction accuracy. To our knowledge, the approach presented in this paper is the first to rely on Petri nets for remaining time prediction of already running cases, which allows us to treat concurrent activities appropriately.

Based on a stochastic model of a business process that can be obtained by expert's knowledge or historical performance information extracted from event logs, this paper introduces a novel prediction algorithm. The prediction algorithm can cope with both non-parametric stochastic models and parametric models of known shape. It is implemented and available as open-source plug-in in the process mining framework ProM [2].

We compare the results of the prediction approach with state-of-the-art approaches [4], and with a model using exponentially distributed durations. As basis for comparison we use a simulated experiment and a case-study based on real container tracking data from a Dutch logistics service provider.

The remainder of this paper is structured as follows. In Sect. 2, we provide preliminary definitions used throughout this work. Subsequently, Sect. 3 describes the proposed method's theoretical background and the implemented solution. Afterwards, in Sect. 4, we evaluate the predictions of different existing approaches to our approach and discuss the findings. In Sect. 5, we compare our work with related methods and highlight conceptual differences. Finally, we conclude in Sect. 6.

2 Preliminaries

In the following, we describe the concepts, on which the prediction algorithm is based. Our goal is to predict the remaining time until completion of a certain business process instance. We use the terms *instance*, and *case* interchangeably in the remainder of the paper.

Basically, the prediction uses historical information (i.e., information on how similar cases have performed in the past), and all the information that we have about the current case. First, the prediction algorithm makes the basic assumption that historical and current process execution information (i.e., the start or end of activities) are stored as events with timestamps. We call a collection of events belonging to a process an *event log*, or simply *log* in the remainder of this paper.

Events are correlated to the activities of a certain case of a process. We group events of a particular case into a *trace* and order the events by the timestamp of occurrence. Note that a trace can be either complete (i.e., the events of a completed case are all in the trace), or still running (i.e., some events are still expected). In order to distinguish these cases and to be able to define the behavior of the process, business process models are widely used in companies [22]. These models serve other purposes as well, e.g., analysis, simulation, ground for common understanding of processes, implementation specification.

In this work, we use business process models to capture performance criteria of a business process. In fact, we use Petri nets as modeling basis in this paper. Petri nets cover the most important control flow constructs of business processes like sequential execution, exclusive choice, parallelism, and synchronization. Furthermore, they have a well-defined semantics and can be verified for correctness [1]. In practice, high-level process modeling languages are common, e.g., the industry standard BPMN, but most of those models can be transformed into Petri nets [15]. We employ an extended Petri net formalism to capture performance aspects, but let us revisit the plain form, first:

Definition 1 (Petri net). *A Petri net is a tuple $PN = (P, T, F, M_0)$ where:*

- $P = \{p_1, p_2, \ldots, p_m\}$ *is a set of places.*
- $T = \{t_1, t_2, \ldots, t_n\}$ *is a set of transitions.*
- $F \subseteq (P \times T) \cup (T \times P)$ *is a set of connecting arcs representing flow relations.*
- $M_0 \in P \rightarrow \mathbb{N}_0^+$ *is an initial marking.*

We restrict the Petri net models to be sound WF-nets, i.e., having an input and output place with every transition on a path between these two places and no dead locks or live locks, cf. [1]. Soundness also requires the output place to be reached eventually from all markings, s.t., no tokens remain in the net. The models do not need to be structured or free-choice, because we use a simulation-based approach to prediction. In this work, we enrich this model with additional stochastic timing information to be able to make predictions of remaining durations.

Definition 2 (GDT_SPN). *A stochastic Petri net with generally distributed transitions (GDT_SPN), is a seven-tuple: $GDT\_SPN = (P, T, \mathcal{P}, \mathcal{W}, F, M_0, \mathcal{D})$, where (P, T, F, M_0) is the basic underlying Petri net. Additionally:*

- *The set of transitions $T = T_i \cup T_t$ is partitioned into immediate transitions T_i and timed transitions T_t.*
- $\mathcal{P} : T \rightarrow \mathbb{N}_0^+$ *is an assignment of priorities to transitions, where $\forall t \in T_i : \mathcal{P}(t) \geq 1$ and $\forall t \in T_t : \mathcal{P}(t) = 0$.*
- $\mathcal{W} : T_i \rightarrow \mathbb{R}^+$ *assigns probabilistic weights to the immediate transitions.*
- $\mathcal{D} : T_t \rightarrow D$ *is an assignment of arbitrary probability distributions D to timed transitions, reflecting the durations of the corresponding activities in the real world.*

GDT_SPN models rely on the well-known notion of generalized stochastic Petri nets (GSPN) by Marsan et al. [17]. The key difference to GSPNs is that we do not enforce the Markovian property, i.e., memorylessness, which restricts transition delays to be exponentially distributed in the continuous case. Typically, real world processes exhibit duration distributions, that are not exponential. For instance, services can contain deterministic time-outs, or durations of activities can belong to normal or log-normal classes, e.g., surgery durations are assumed to follow a log-normal distribution [21]. Accordingly, we allow to use *any* parametric or non-parametric distribution, as long as we can draw samples from it. In comparison to GSPNs, which have the convenient property that they are isomorphic to Markov processes, we lose the possibility to calculate expected durations analytically and efficiently, but resort to analysis by Monte Carlo simulation instead [6].

When lifting the Markov assumption, an execution semantics needs to be selected [16]. We use race-semantics with enabling memory, as defined in [16] and used in [25], too. Firing rights between concurrently enabled immediate transitions are resolved probabilistically based on their weights. When immediate and timed transitions are enabled, immediate transitions fire first, due to higher priority. When only timed transitions are enabled, they race for the right to fire first. This allows to model time-outs. The enabling memory semantics specifies that concurrent non-conflicting activities do not lose their progress when another transition fires. However, if a transition gets disabled, it has to restart its work the next time it becomes enabled. The class of GDT_SPN models allows to model most reasonable business processes with their specific timing properties.

Now, how do we obtain GDT_SPN models with duration distributions for every transition? If the process models to be enriched with performance information are not known in advance, algorithms from process mining can be used to infer the Petri net models that capture the observed behavior in the log [5,2].

In previous work [19], we described a method based on replaying event logs on Petri nets using the notion of alignments [3]. By aligning the historical traces to the model and by calculating activity durations for each transition, we gain statistical timing information. Statistical information collected in this way can be used to fit statistical parametric distributions (e.g., normal, log-normal, exponential, phase-type), and non-parametric distributions [11]. Non-parametric techniques might prove more accurate, if the durations contain irregularities, such as two peaks, but are also subject to overfitting.

Preferably, in latter cases, it would help to find the reasons for such irregularities, and thus be able to separate cases belonging to one heap from those belonging to the other heap. Machine learning methods, e.g., classification algorithms, are frequently used for this task. Example applications are described in [7], where the authors used non-parametric regression techniques on case attributes such as the amount of the insurance claim that proved well as indicator for case duration. However, these additional approaches are out of scope of this work, and we focus on the case, where no additional information is present in the log.

3 Prediction during Process Execution

In this section the core contribution of this paper is introduced and the prediction algorithm is presented. Given the preliminaries described in the previous section, we first introduce the key concept that the prediction is based on, i.e., constrained activity durations.

3.1 Constrained Activity Durations

An important assumption for our work is that the system detects process relevant events if they happen—i.e., the events are always recorded in the system—and that the time from occurrence to detection is negligible. This assumption applies especially for processes that are supported by process aware information systems. It allows us to use information that goes beyond the mere detection of observed events. We can use the information that an enabled activity has *not been completed yet* at a given point in time, if the event for completion of the task was not observed yet.

The concept is illustrated with an example. Consider the GDT_SPN model of an insurance claim process in Fig. 1. In this process, there are two options for proceeding after the *Check case* activity. Either the case will be handled with a 70% chance or it will be rejected in the remaining 30% of the cases. In the model, these probabilities are depicted as annotated weights of the immediate transitions, shown as black bars competing for one token. The delay distributions are specified in a parametric form in day units, e.g., the *Check case* transition is exponentially *(exp)* distributed with a firing rate parameter λ=2 days, and the *Handle case* transition is normally *(norm)* distributed with a mean of five days and a standard deviation of one day. After either the handling or the rejection of the case, this simple process is completed, i.e., the token reaches the output place.

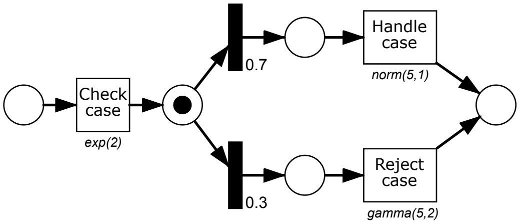
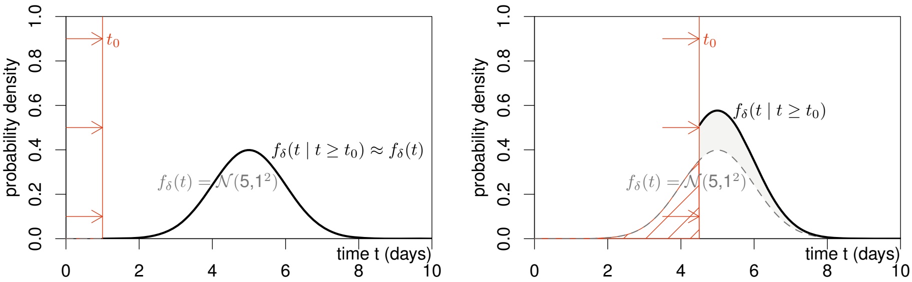


Fig. 1. Model of a simplified insurance process. The current trace contains only the event indicating that activity *Check case* has finished.

In the current situation depicted in Fig. 1, the case was checked and awaits a decision. Without loss of generality, we assume the upper branch was chosen and align the time axis of the density function of the duration of the *Handle case* transition to be zero when it becomes enabled.



(a) After one day, the truncated density is almost unchanged

(b) After 4.5 days, the truncated density is significantly different from the original density

Fig. 2. Truncated probability density functions of the duration of activity *Handle case*, i.e., constrained to (a) being greater than one day, and (b) being greater than 4.5 days.

Figure 2 shows the resulting distribution of the *Handle case* transition at two *later* points in time. Fig. 2(a) depicts the normally distributed probability density, which is the well-known bell-shaped curve, where after one day, activity *Handle case* was not yet completed. Note that this observation does not change the original distribution significantly, as it is unlikely that the activity is completed earlier than after one day.

Fig. 2(b) shows the the same situation, but more time has passed without detecting the end of the activity. Let $\delta\in\mathcal{D}$ be the duration distribution of the *Handle case* activity. In dashed grey the probability density function of the original duration $f_\delta(t)$ is depicted. The vertical line shows the current time t_0 that advances from left to right, as time proceeds. The thick black curve $f_\delta(t \mid t \geq t_0)$ is the truncated density of the remaining cases. It depicts the conditional probability density of the duration excluding the non-consistent cases (crossed out area left from t_0).

More generally, let $\tau \in T_t$ be a timed transition with the assigned duration distribution $\delta = D(\tau)$. Then, $f_\delta(t)$ is the probability density function and $F_\delta(t) = \int_{-\infty}^{t} f_\delta(t)\mathrm{d}t$ is the probability distribution function of the duration of τ. Let TD be the time domain. Let $t_0 \in TD, t_0 \geq 0$ be the current time since enabling of τ. Let further δ_{Dirac} denote the Dirac delta function which captures the whole probability mass at a single point. Then we define the density function of the truncated distribution as:

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

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 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. From that distribution, 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 already completed the running activity before the current time. In contrast to using conditional probability density functions, traditional methods predict the remaining duration of a case only on event arrival, and subtract elapsed time from the predicted duration at later points in time.

Note that the presented approach is improving the prediction of running activity durations. Therefore, it is most effective in processes, where some activities have a relatively high impact on the process duration.

3.2 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.

In order to make predictions for a single case, the current state of the case and the model that captures experiences about the behavior of the process needs to be known. The prediction algorithm takes four inputs: (1) the *GDT_SPN model* of the business process, cf. Definition 2 in Sect. 2, (2) the current *trace* of the case, i.e., all observed events up to time t_0, (3) the *current time* t_0, and (4) the number of simulation *iterations* indicating the precision of the prediction. Algorithm 1 describes the procedure.

The algorithm is straightforward. 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, an extension based on techniques from [3] can be used to align non-fitting traces with the model.

Algorithm 1. Prediction algorithm

```
procedure PREDICT(model, trace, current_time, iterations)
    currentMarking ← replay(trace, model)    ▷ replay the observed events in the model
    times ← new List()    ▷ used to collect results
    for all $i \in iterations$ do
        time ← simulateConstrained(model, currentMarking, current_time)
        times.add(time)
    end for
    return getMean(times)    ▷ the average of the simulated values
end procedure
```

As a second step the algorithm collects simulation results, i.e., completion *times*, of a given number of simulation iterations in a list. Each simulation run represents a sample from the possible continuations of the process according to the model. 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 Sect. 3.1. The completion times of all simulated continuations of the case are collected and the algorithm returns the mean of these sample values.

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 ± 3 percent of the predicted value.

3.3 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. 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 used only 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.

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

In our case, rejection sampling becomes inefficient, if the current time has progressed beyond most of the distribution's probability mass, as almost all samples will be rejected. For such cases, we also implemented a slice sampling method that samples directly from the conditional probability density $f_\delta(t \mid t \geq t_0)$ in Equation 1 by a random walk under the density function. Slice sampling can be used to sample from any distribution as long as we can compute the corresponding density function.

4 Evaluation

In the following evaluation, we analyze the quality and run time of our approach. To assess prediction quality, we compare our approach to the related prediction method described in [4], and an analytical method based on regular GSPNs. We further investigate how the presented approach scales in terms of run time for different model sizes. We start with the evaluation of the prediction quality. Therefore, we use both a simulated model, and real data from a logistics provider in the Netherlands. First, we explain the experimental setup.

4.1 Experimental Setup

The experimental setup is depicted in Fig. 3. In the synthetic cases a GDT_SPN model that contains both the control flow structure definition and the performance specification in form of duration distributions is used. From this model 10000 traces of execution are generated and stored in the simulated log. Besides the simulated log, also the Petri net model (i.e., the underlying Petri net of the GDT_SPN model), is used as input. In real settings, we have a Petri net and a log given as input. To evaluate the prediction quality, a 10-fold cross validation is performed. Therefore, the log is split into ten evenly divided parts and nine of them are used as the training log to learn the performance behavior and the remaining part as test log to test the prediction accuracy. We iterate over these parts, such that each of them is used once as test log. The Petri net is enriched to a GDT_SPN by collecting the performance data of the training log, as described in [19].

We are interested in the different prediction methods' accuracy to predict the remaining duration of a case *any time* during the process. Therefore, we trigger the predictions

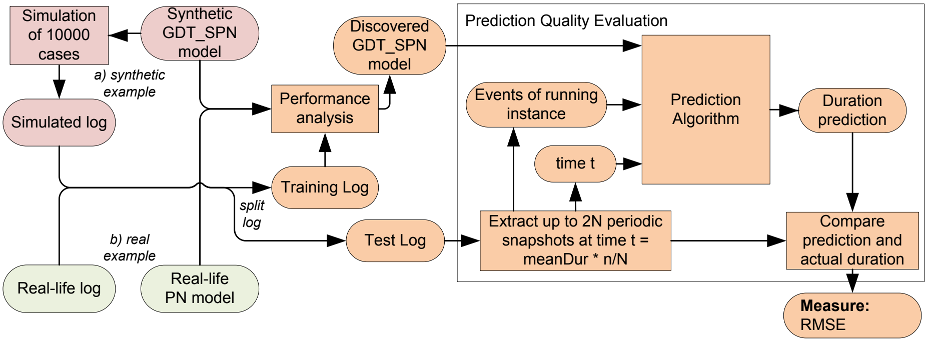
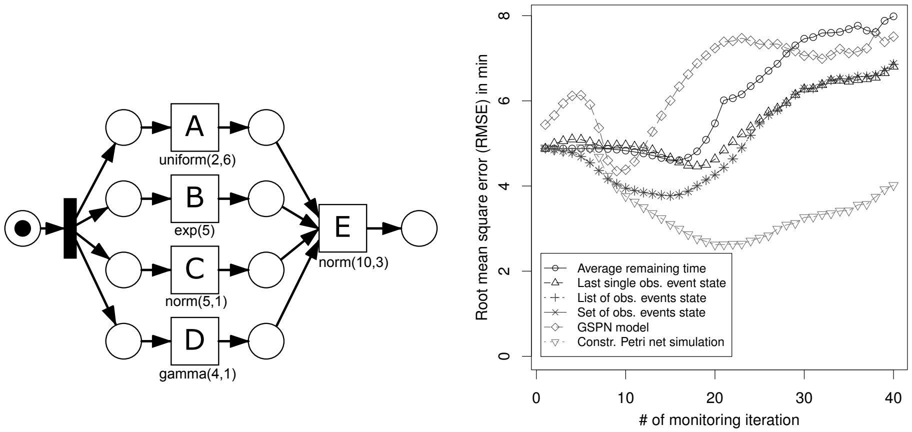


Fig. 3. Experimental setup for evaluating the prediction quality of the algorithm



(a) A GDT_SPN process model with four parallel activities A, B, C, D and a final activity E, with annotated duration distributions.

(b) Root mean square errors (RMSE) in minutes at 40 periodic predictions. Mean duration: 17.32 minutes.

Fig. 4. A model with four parallel branches (a) and corresponding prediction errors (b) using 10-fold cross-validation with 40 periodic predictions, s.t. the 20th iteration is at the mean duration

periodically. The length of such a period is based on the mean process duration that is obtained from the training log. More precisely, $2N$ periodic predictions are made for each instance in the test log, such that the Nth snapshot is at the mean duration of the process. This means that while initially all instances still run, some instances will be finished at later prediction iterations. Note that only predictions for still running cases are added to the resulting statistics.

The evaluation proceeds for each trace in the test log as follows. At each iteration of the periodic prediction, we compute the relative time t_0 to the trace start and pass t_0 and the partial trace containing the events of the case with time $\leq t_0$ to the prediction algorithm described in Sect. 3. The predicted duration for different prediction methods are computed and compared to the actual duration of the trace from time t_0. The simplest method for prediction is using the *average remaining time*, which is simply the mean process duration gathered from the training set minus the elapsed time t_0. Additionally, we compare against the state transition systems approach [4] with different configurations. Predictions based on the history sharing (i) the *last observed event only*, (ii) the *list of all previous observed events*, and (iii) the *set of all previous observed events*. Finally, we also compare our approach against a regular GSPN based approach, i.e., an approach where only exponential distributions are allowed in timed transitions of the model.

Note that if any of the methods predicts a negative remaining time, i.e., that the current case should have completed already at time t_0, the predicted duration is set to 0.

4.2 Results of a Simulated Experiment

Figure 4 shows (a) an example model containing four parallel branches that is used for the simulated experiment, and (b) the root mean square error (RMSE), for each of the 40 periodic predictions mentioned above. The RMSE is an error measure for quantifying the error between a predicted and a real value, cf. [9]. The smaller the RMSE, the better is the prediction in average. All prediction algorithms perform similarly at the start of an instance, except the GSPN model which fits exponential distributions to all timed transitions. Although the prediction quality of the GSPN model is worse than that of the comparison models, it can be analyzed efficiently. After some time has progressed, however, the prediction based on the constrained Petri net simulation, introduced in Sect. 3, outperforms the other approaches significantly.

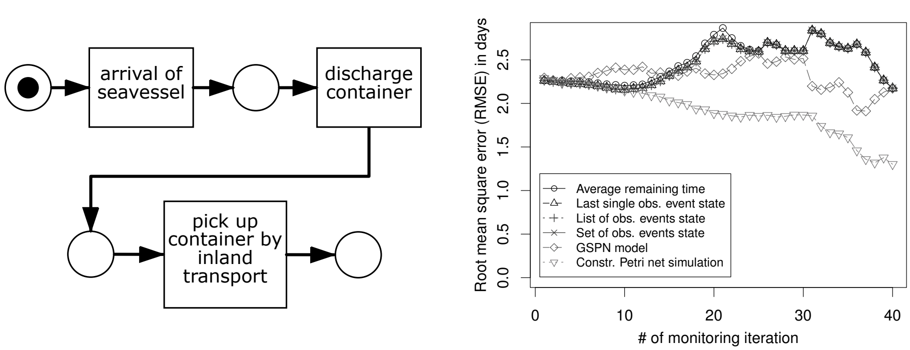
4.3 Results of an Industry Case Study

We conducted the experiment described in Sect. 4.1 also for real data from a logistics provider in the Netherlands. Fig. 5 shows (a) the process model, and (b) the prediction errors side by side. The event log of the logistics process contains entries for arrival of a seavessel, discharge of the container, and the date of picking up the container by inland transport in a sequential order. It contains 784 cases over the year 2011.

In this case study, the different abstraction levels of the comparison method [4] collapse to the same results, cf. Fig. 5(b), because the process is sequential. The GSPN method yields comparable results as the benchmark methods in [4]. But we can observe that our presented approach produces more accurate predictions of the remaining process duration than the benchmarks.

To summarize the results of the evaluation of prediction quality, we highlight the following characteristics of our approach:

- At early prediction periods the approach performs about as well as the benchmarks.
- The improvements become more significant as time proceeds.
- The approach is most valuable for long running cases, which are critical to be detected and avoided.



(a) Model of a sequential logistics process capturing the arrival of seavessels, container discharges and further inland shipping.

(b) RMSE in days for the logistics process at 40 periodic predictions. Mean duration: 4.03 days.

Fig. 5. Model (a) and prediction errors (b) for the logistics provider case study using 10-fold cross validation with 40 periodic predictions, s.t. the 20th iteration is at the mean duration.

4.4 Scalability Analysis

We propose to use simulation as means 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., *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:

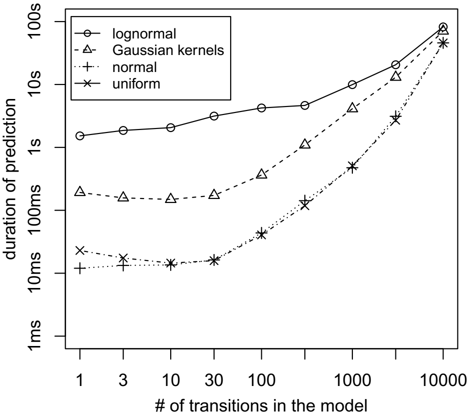


Fig. 6. Prediction durations for differently sized randomly generated, acyclic GDT_SPN models. Axes in log-scale.

- 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 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. Regarding *computing power*, we used a regular laptop computer with a Pentium i7 620M (2.66 GHz cores) equipped with 8GB of ram. We varied the other two factors, i.e., the *average number of transitions* and the *kinds of distributions* used. Fig. 6 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 medium sized model (100 transitions) takes around 300 milliseconds for rather expensive non-parametric Gaussian kernel density estimation. Prediction of models with lognormally distributed values takes long because of higher variance. Note that a prediction of a model with 10000 transitions still is feasible in less than 100 seconds with these configurations.

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 implemented in [25].

5 Related Work

A lot of related work exists that deals with prediction based on historical observations. Common are predictions based on time-series data, e.g., for the stock market or for sales numbers of a company. Whereas in time series data, the individual values tend to depend strongly on the previous values, in a business process, these dependencies are less common. Therefore and due to space limitations, we refer to the overview of existing forecasting methods in [9]. Work on the analysis of trends and change points in *processes* can for example be found in [23]. However, for this work, we assume that the current model of the process performance is representing the current real world performance. So either the process is in a steady state, or mentioned methods, as in [23], are used to keep the model up to date.

5.1 Process Related Predictions

There has been work on prediction of case durations based on historical observations. In their work, van der Aalst et al. use the available information in logs to predict the remaining duration based in observed durations in the past [4]. They create an annotated state transition system for the logs, which can be calibrated in terms of abstraction, and collect remaining durations for each state from the log. 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 [4] serves as one of the benchmarks for our prediction method in the next section.

Building on the work in [4], Folino et al. [8] present an improvement based on predictive clustering. They make use of additional contextual information of a trace, e.g., the current workload, to perform clustering. The idea is to group similar traces and base predictions on such subsets of the log for new ones with similar features. They use the predictions to warn in case of a predicted transgression of a threshold. Our prediction method could substitute the state-transition based method in [8], improving the results for processes with parallelism, and during execution of the instances.

Other work for prediction of performance was presented by Hwang et al. [10] and similarly Zheng et al. [24]. 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. 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 also consider already running instances.

The work presented by Leitner et al. [14] considers already running instances, too. 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. Similarly, Pika et al. [18] 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. 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. Kang et al. [13] advocate also business process monitoring in real-time. Their approach is based on classifying historical traces in correct and incorrect traces by inductive data mining techniques, e.g., support vector machines. Similarly as in our motivation, their goal is to predict and classify current instances, e.g., if they are likely to produce failures. However, their approach is only capturing sequential processes, and event's timestamps are not considered in the prediction.

Simulation has also been proposed and used for operational decision making by Rozinat et al. [20]. 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 stochastic Petri nets with generally distributed firing times, 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 [25]. However, those works rather focus on transient analysis, e.g., average throughput and waiting times, instead of predicting remaining durations of single instances with conditional probability densities based on the current prediction time.

5.2 Quality of Service Related Forecasting

Jiang et al. [12] handle time series in business activity monitoring and focus on detecting outliers and change points. Their approach does not consider the process structure, but they only consider specific features of a process, such as customer usage profiles, and within those features they focus on adapting the prediction model to observed trends and changes. Another approach by Zeng et al. [23] applies the ARIMA forecasting method to predict performance criteria for event sequences (corresponding to *traces* in our terminology) and thus support seasonality of changes. They do not use that approach for single instances, but rather for aggregated key performance indicators (KPIs). Their prediction can be mapped to the prediction approach in [4] with the states distinguished as lists of ordered events. Based on that model the values are classified by regression to separate them in those that meet the KPIs, and those that violate them. There is a drawback to using event sequences for prediction in processes with parallelism, as there is a combinatorial state space issue with the interleavings, such that for a single prediction much otherwise useful training data are spread to other interleaving sequences and do not influence the prediction for the current sequence. Compared to that, our work improves on the aspect of prediction of a single case in *real-time* and is less likely to suffer from sparse training data issues in processes with parallelism.

Trend aware forecasting methods are out of scope of this paper, because we assume the process to be in steady state. If the process performance is subject to seasonality or trends, an integration of methods like ARIMA seems promising, however. To the best of our knowledge, the main contribution of this work, i.e., using temporal restrictions to make predictions more accurate, has not been proposed in literature before.

6 Conclusion

We described a relevant and common setting for the prediction of remaining process duration, where a process aware information system is aware of changes in process instances immediately. Based on this setting, we isolated one important aspect, i.e., the conditional duration distributions based on passed time and investigated the effects on prediction accuracy. We compared our results with state-of-the-art approaches devised for similar settings, and with the results of efficiently analyzable GSPN models.

The presented prediction approach is able to capture parallelism in business processes naturally, because it is based on Petri nets, as opposed to techniques based on state transition systems [4,8]. Another advantage of Petri nets is that they are able to capture resources in a native way. As future work, we want to integrate the resource perspective to make the prediction account for dependencies between instances of the process.

Further, we plan to lift the independence assumption and take correlations between activity durations into account. Classification based on case attributes has shown much potential to improve predictions, cf. [8], and we expect similar gains in prediction accuracy when combined with this method. In this paper, we assumed the model to represent the current state accurately, which in reality is often not the case, because the process might be subject to changes, e.g., changes due to seasonality, resource situation or process redesign. The integration of a mechanism to detect such changes and adapt the model accordingly would further increase the accuracy of predictions.

References

1. van der Aalst, W.M.P.: Verification of workflow nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997)
2. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
3. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2, 182–192 (2012)
4. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Information Systems 36(2), 450–475 (2011)
5. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
6. Bobbio, A., Telek, M.: Computational restrictions for SPN with generally distributed transition times. In: Echtle, K., Powell, D.R., Hammer, D. (eds.) EDCC 1994. LNCS, vol. 852, pp. 131–148. Springer, Heidelberg (1994)
7. van Dongen, B.F., Crooy, R.A., van der Aalst, W.M.P.: Cycle time prediction: When will this case finally be finished? In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part I. LNCS, vol. 5331, pp. 319–336. Springer, Heidelberg (2008)

8. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 287–304. Springer, Heidelberg (2012)
9. De Gooijer, J.G., Hyndman, R.J.: 25 years of time series forecasting. International Journal of Forecasting 22(3), 443–473 (2006)
10. Hwang, S.Y., Wang, H., Tang, J., Srivastava, J.: A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. Information Sciences 177(23), 5484–5503 (2007)
11. Härdle, W.: Applied nonparametric regression. Cambridge University Press (1990)
12. Jiang, W., Au, T., Tsui, K.L.: A statistical process control approach to business activity monitoring. IIE Transactions 39(3), 235–249 (2007)
13. Kang, B., Kim, D., Kang, S.H.: Periodic performance prediction for real-time business process monitoring. Industrial Management & Data Systems 112(1), 4–23 (2011)
14. Leitner, P., Wetzstein, B., Rosenberg, F., Michlmayr, A., Dustdar, S., Leymann, F.: Runtime prediction of service level agreement violations for composite services. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 176–186. Springer, Heidelberg (2010)
15. Lohmann, N., Verbeek, E., Dijkman, R.: Petri net transformations for business processes – A survey. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 46–63. Springer, Heidelberg (2009)
16. Marsan, M.A., Balbo, G., Bobbio, A., Chiola, G., Conte, G., Cumani, A.: The effect of execution policies on the semantics and analysis of stochastic Petri nets. IEEE Transactions on Software Engineering 15(7), 832–846 (1989)
17. Marsan, M.A., Conte, G., Balbo, G.: A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. ACM TOCS 2(2), 93–122 (1984)
18. Pika, A., van der Aalst, W.M.P., Fidge, C.J., ter Hofstede, A.H.M., Wynn, M.T.: Predicting deadline transgressions using event logs. In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBIP, vol. 132, pp. 211–216. Springer, Heidelberg (2013)
19. Rogge-Solti, A., van der Aalst, W.M.P., Weske, M.: Discovering stochastic Petri nets with arbitrary delay distributions from event logs. In: BPM Workshops. Springer, Heigelberg (to appear)
20. Rozinat, A., Wynn, M.T., van der Aalst, W.M.P., ter Hofstede, A.H.M., Fidge, C.J.: Workflow simulation for operational decision support. Data & Knowledge Engineering 68(9), 834–850 (2009)
21. Strum, D.P., May, J.H., Vargas, L.G.: Modeling the uncertainty of surgical procedure times: Comparison of log-normal and normal models. Anesthesiology 92(4), 1160–1167 (2000)
22. Weske, M.: Business Process Management: Concepts, Languages, Architectures, 2nd edn. Springer (2012)
23. Zeng, L., Lingenfelder, C., Lei, H., Chang, H.: Event-driven quality of service prediction. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 147–161. Springer, Heidelberg (2008)
24. Zheng, H., Yang, J., Zhao, W., Bouguettaya, A.: QoS analysis for web service compositions based on probabilistic QoS. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 47–61. Springer, Heidelberg (2011)
25. Zimmermann, A.: Modeling and evaluation of stochastic Petri nets with TimeNET 4.1. In: 2012 6th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS), pp. 54–63. IEEE (2012)