Time Series Petri Net Models[⋆]

Enrichment and Prediction

Andreas Rogge-Solti[1], Laura Vana[2], and Jan Mendling[1]

[1] Institute of Information Business
[2] Institute for Statistics and Mathematics
Vienna University of Economics and Business, Austria
{andreas.rogge-solti,laura.vana,jan.mendling}@wu.ac.at

Abstract Operational support as an area of process mining aims to predict the temporal performance of individual cases and the overall business process. Although seasonal effects, delays and performance trends are well-known to exist for business processes, there is up until now no prediction model available that explicitly captures this. In this paper, we introduce time series Petri net models. These models integrate the control flow perspective of Petri nets with time series prediction. Our evaluation on the basis of our prototypical implementation demonstrates the merits of this model in terms of better accuracy in the presence of time series effects.

Keywords: Predictive analytics, business intelligence, time series, Petri nets

1 Introduction

The analysis of business processes is of growing importance to companies for managing their operations and for tailoring effective process-aware information systems. The amount and detail of available data on business processes has substantially increased with a more intensive usage of information systems in various domains of business and private life. Process mining techniques make use of such data in facilitating automatic discovery, conformance analysis and operational support based on log data of actual process executions [2].

While discovery and conformance have been intensively studied recently, there is a notable gap of work that approaches operational support from a time prediction perspective. The few examples in this area include a performance prediction model that captures levels of load as a context factor [8], queuing networks to model business processes with waiting lines [23], or time prediction based on transition systems and log data [4]. On the other hand, it is well established that business process performance is often influenced by periodic effects, trends and delays that can range from intra-day variance of task performance of a process participant to storm season in Australia multiplying lodged insurance claims [3]. Up until now, there is no model that is integrating such effects with the control flow of a process.

⋆ This work was partially supported by the European Union's Seventh Framework Programme (FP7/2007-2013) grant 612052 (SERAMIS)

Against this background, we introduce a formal model that combines Petri nets with the analytical power of time series analysis. In this way, we are able to directly represent periodic effects, trends and delays together with the control flow specification. Our model can be described as a specific kind of a stochastic Petri net, in which the distribution and weight of each transition is replaced with time series models. The formalism is flexible, in that it can use very simple models, for example the average of the durations, or also (if applicable), time series models with seasonality and trend components. We extensively evaluate this model in synthetic settings and with a case study from real-life.

The remainder of this paper is structured as follows. Section 2 presents an introductory example and summarizes prior research on stochastic Petri nets. The formal model with its semantics and the methods to enrich it is presented in Section 3. Then, in Section 4, we present the evaluation setting and results. Finally, we conclude in Section 5 and outline challenges for future research.

2 Background

In the following, we discuss an illustrative example to motivate the need for models that can capture time series and discuss required formalisms.

2.1 Illustrative Example

It has been widely acknowledged that business processes are subject to seasonality [4] and effects of delaying [23]. Such effects can be modeled as time series. Figure 1 shows a respective time series for the number of airline passengers per month in thousands. The data is from the textbook by Box et al. [5]. It can be seen that there is a trend of increase which is mixed with seasonal effects and delays.

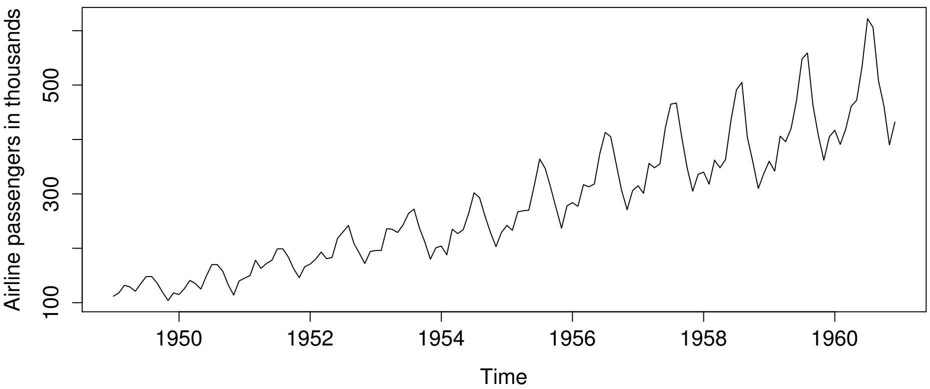


Figure 1: Monthly airline passenger counts in thousands [5]. The data shows a clear trend and also a yearly seasonal component can be observed.

Consider now a travel agency that operates a call center to handle such airline passenger bookings. A corresponding Petri net model is depicted in Figure 2. It shows a call center process from the view of the customer calling. Places are depicted as circles and transitions as boxes. We have two kinds of transitions in this model. The transitions depicted as white correspond to process events (e.g., a customer call is received, the voice receiving unit is left, the service ended). These transitions signal a change in the process state and correspond to progress of the case. The grey transitions are *invisible* to the system. When a customer calls, the voice receiving unit takes the call and provides routing to the corresponding service station. The customer can hang up, or be routed forward. Depending on whether the service station is busy, the client needs to enter a queue first. If the client is tired of waiting, she can hang up. Or finish waiting in the queue to be served. At the service station the client is connected to a service employee and finally when the service is finished, the process completes. To better predict the duration of the process, we need to capture time series effects (e.g.,seasonality) within the process model.

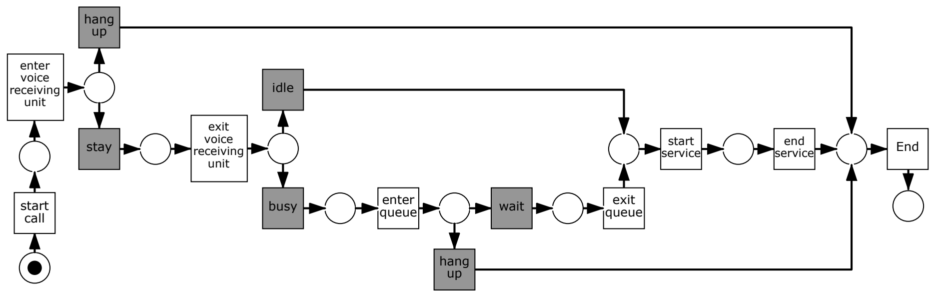


Figure 2: Call center process as a Petri net.

Throughout this paper, we will use the Petri net formalism for modeling and prediction of business processes, more precisely the specific class of *workflow nets* [1]. We always assume the workflow net properties in this paper, and use the term Petri net instead. Petri nets are a versatile tool that allow us to capture behavioral relations, like concurrency, conflict and sequential behavior, as well as to capture repeated cycles in a process in a compact form. Essentially, all important process modeling languages, no matter if *imperative* like BPMN, EPC, or UML Activity Diagram [14], or *declarative* such as Declare models [19] can be mapped to Petri nets. Therefore, by building on Petri nets, we effectively offer a means to predict durations for any model that has a Petri net representation.

2.2 Petri Nets and Time

Even though there has been extensive research on combining Petri nets with time, there is up until now no model available that directly integrates it with time series characteristics. However, various extensions to Petri nets have been proposed to capture the non-functional properties—like the performance—of systems.

One of the first extensions in this direction was to enrich each transition in a Petri net with exponential firing rates and the resulting models are called *stochastic Petri* *nets* (SPN) [15]. These models are memory-less in their firing behavior, that is, their behavior is independent of the time spent in a certain state. This property makes SPN isomorphic to Markov chains [18]. To overcome this simplification, *non-Markovian* stochastic Petri nets were proposed, which allow for more general modeling of the duration distributions of transitions [9]. These models usually assume independence of durations within a process instance, and also between cases. An extension to capture dependence on the history of the current case was introduced in the notion of History dependent stochastic Petri nets [22]. Latter models allow us to capture an often encountered phenomenon in business processes with cycles: the probability to leave a cycle increases with each iteration.

Besides Petri net based models, more abstract models building on transition systems were also proposed to predict remaining process durations [4]. These type of approaches extract a state s from the given observed process trace, and predict the average remaining durations of former cases that also passed through state s. Extensions to make these methods more accurate have been devised, for example to *cluster* cases based on the system load (i.e., the number of currently active process instances of a process) [8]. Another approach to predict the remaining time is based on feature-based regression of different characteristics of the case and works well, if the features are correlated to the remaining duration [6]. These methods work well, if process data is available to the process engine, and an extension of our approach with regression is certainly worthwhile to investigate, but out of scope for this paper.

All these models, however, do not explicitly take into account existing correlations between cases, and are unable to capture seasonality and trends in data. To our knowledge, we present in the following the first work integrating time series and Petri nets.

2.3 Time Series

Time series data arise naturally when monitoring processes over a given period of time. Time series analysis methods have the advantage of accounting for the fact that data observed over time might have an underlying internal structure. Hence, the goal of time series analysis is the understanding of this underlying structure and of the forces driving the observed data as well as forecasting of future events. More formally, a time series is defined as a sequence of observations at given times. We use the following notation to describe the past N observations: $y_1, \dots, y_N$. Further, we are interested in the value of a time series h steps ahead in the future, that is, we predict y_{N+h}. The parameter h is called *horizon*, as it marks how far we would like to predict into the future.

Observed time series data can be decomposed into several potential components: a random or shock component, a trend component (a systematic linear or non-linear tendency in the series), or a seasonal component (patterns that repeat themselves in systematic time intervals). Other patterns in time series data might include autocorrelation (correlation with different lags of the data), also known as autoregressive (AR) processes, correlation with different lags of the shocks, called moving average (MA) processes, or both. Latter are called ARMA processes. One property necessary for predicting and modelling time series is stationarity, which requires a constant mean of the series. In general, non-stationary data is transformed to stationary by differentiation.

There are different techniques for modelling and forecasting time series data [10]. The most popular ones include Exponential Smoothing and the Box-Jenkins ARIMA (AutoRegressive Integrated Moving Average) models, which except for incorporating AR and MA patterns can also account for seasonality components. There are also naive approaches to forecasting, e.g., simply using the last observed value of the current time series as forecast. The large body of research dealing with forecasting for time series is concisely summarized in the review by de Gooijer and Hyndman [10].

In the following, we will denote as $\mathbf{Y}$ the universe of time series models, i.e., models that when provided with a given time series $\{y_1, \ldots, y_N\}$ can be used to generate a prediction for the given forecast horizon. Note that we will not restrict the kind of models that can be used to certain kinds of time series models. In fact, in the evaluation, we will compare different approaches, from naive predictors to the automatic selection of a fitting ARIMA model.

3 Time Series Petri Nets

In this section, we describe the underlying model that we propose for encoding seasonality and trends in prediction of business process durations.

3.1 Definition and Semantics

In contrast to previous methods using Petri nets for prediction, we allow the model to encode correlations to previous observed values at given stations in the process, i.e., at the transitions. In this sense, the model we propose builds on the one of Schonenberg et al. [22], into which we integrate time series concepts such as correlations to previous instances which passed through the part of the model. This allows us to capture seasonality in durations and decisions, e.g., at Christmas, more customers choose the gift wrapping option in the order process. The choice between conflicting transitions is captured as weights which depend on time and on the previous trends or seasonal patterns that can be observed.

Given a plain model of a Petri net model as $PN = (P, T, F, M_0)$, the time series Petri net (TSPN) is a model is defined as follows:

Definition 1 (Time Series Petri Net). *A TSPN is a six-tuple:* $TSPN = (P, T, F, M_0, C, \mathcal{D})$*, where* (P, T, F, M_0) *is the basic underlying Petri net.*

- P *is a set of places.*
- *The set of transitions* $T = T_I \cup T_T$ *is partitioned into immediate transitions* T_I *and timed transitions* T_T
- $F \subseteq (P \times T) \cup (T \times P)$ *is a set of connecting arcs representing flow relations.*
- $M_0 \in P \to \mathbb{N}_0$ *is an initial marking.*
- $C : T_I \to \mathbf{Y}$ *assigns to the immediate transitions their corresponding time series model that represents their firing rate (which can vary over time).*
- $\mathcal{D} : T_T \to \mathbf{Y}$ *is an assignment of time series models to timed transitions reflecting the* durations *of the corresponding process states.*

This definition of TSPN models is aligned with the well-established generalized stochastic Petri net (GSPN) [17] model, where immediate transitions are responsible for routing decisions, and timed transitions represent how long it takes to proceed to the next step in the process. The execution semantics of the TSPN model can be chosen from the combinations of conflict resolution and firing memory policies, as it is also available for non-Markovian stochastic Petri nets [16]. Without loss of generality, we assume that conflicts between immediate transitions are resolved probabilistically with respect to their estimated firing rates as forecast by their time series, and conflicts between concurrently enabled timed transitions are resolved by a race policy, that is, the fastest transition fires first. Those transitions that lose a race can keep their progress (i.e., we use the enabling memory policy) until they get disabled. Note that TSPN models can be used, like other stochastic Petri net formalisms to estimate remaining times between two points in a process, or the chance that a certain event occurs. The prediction is based on the successive prediction and combination of activity durations and decision probabilities.

3.2 Challenges of Enriching Time Series Petri Nets

The enrichment process is closely following the algorithm as described for generally distributed transition stochastic Petri nets [20]. In a nutshell, the event log that contains the collected execution information of a process is replayed on the Petri net model. This Petri net can be either manually provided, or discovered from the event log by process mining techniques [2]. During replay, we obey the semantics of the TSPN model, which are selected by the user. The result is an enriched Petri net, where for each transition we collected the *durations* from enabling to firing with the associated timestamp of firing.

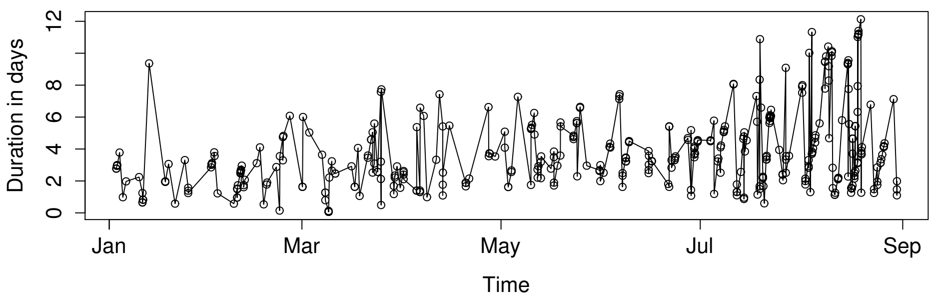


Figure 3: Irregularly spaced time series of a logistics process. The duration until a container is picked up is depicted.

Various challenges and modeling options have to be considered for the construction of an appropriate model. We encountered the following challenges for enriching Petri net models to TSPN models.

Immediate Transitions Without prior knowledge, transitions in Petri net models can be either immediate or timed transitions. Only by careful analysis of the durations can we decide whether a transition is immediate.

Irregularly Spaced Observations Note that in contrast to common time series data, we consider durations of activities (or more generally durations of certain process states), where the gathered observations are irregularly spaced. See Figure 3, which shows the durations for transport containers remaining at a harbor. In this case, the density of the collected observations clearly demonstrates that we collected more data points in August than in March.

Outliers It can happen that the duration of an activity is a rather extreme value compared to the other values. There can be many possible reasons for outliers—e.g., coordination problems of process participants, rare cases that require much more work than normally. Because outliers can negatively impact the accuracy of learned models, it is often necessary to remove outliers first, before learning the model parameters of a time series from data.

Decision Probabilities In business processes that capture choices with regard to the following path in the process, decisions can be modeled as probabilistic choices. We need to find a way that allows us to capture temporal patterns and dependencies in the decision probabilities.

Hidden Patterns and Model Selection It is no trivial task to identify the appropriate model that fits the observed data well and also generalizes to future data. Sometimes, it is surprising how complex models with many parameters are fitted to data, but do not generalize well to new observations. The difficulty lies in the balance between *overfitting* and *generalization* [11].

Negative Values Time series models are usually agnostic of the sign of the data (be it positive or negative). In our setting, we have durations and firing rates of transitions, which need always be positive.

Besides these challenges, we need to mention an important technical detail, as it might be easily missed when constructing time series from collected data: The collected data is recorded when the transitions *fire*. However, as the model should represent the duration of a transition in dependence of the time, we need to shift the observation to the point in time when the state was *entered*.

3.3 Design Decisions for the Enrichment of Time Series Petri Nets

In this section, we discuss possible solutions to these challenges, and the solutions we chose to implement for a prototypical evaluation.

Detecting Immediate Transitions The solution we chose to identify immediate transitions is to check whether the 95th percentile of the collected values is below a given (small) threshold. This way, the method is robust to minor differences in system times in distributed settings, and also robust to up to 5 percent outliers in the data.

Avoiding Irregularly Spaced Observations We apply a straight-forward technique to convert irregularly spaced time series into equidistant observations, which is *aggregation* to a coarser grained time unit. For example, one can aggregate the durations of a given activity to an hourly basis using the average of the observations in each hour as the aggregate value. By this transformation, seasonal patterns are easier to identify, as the averages of each morning at 10 am are 24 observations apart, with a weekly period of $24 \cdot 7 = 168$.

Removing Outliers and Missing Values One way to deal with outliers is to remove them from the training data to which we want to fit the time series models. There exist ways to detect temporal outliers in business processes [21], which we could use to identify a certain number of the most extreme values. Further, there also exists an implementation in `R`, which we use to remove outliers and missing values from time series data.[3]

Using Time Series to Model Decision Probabilities Our goal is to keep the model consistent, i.e., not only allow for durations of activities to be dependent on time, but also to allow *decision probabilities* to be variable over time. Therefore, we do not simply count the number of times a certain decision was taken in comparison to the conflicting decisions, but capture the *count* of transition firings as the time series. Let us consider a case of two conflicting transitions. By aggregating the counts on an hourly basis, we can determine the firing rate of these transitions in the next hour as the ratio between the two forecasts of the corresponding time series models. Thereby, we can effectively capture temporal patterns (e.g., seasonal components, trends) in the decision probabilities of TSPN models.

Identifying Hidden Patterns and Selecting Models To create a plausible prediction model, usually we need to integrate domain knowledge of experts, who know the business processes well. There is no silver bullet solution for modeling the data, although the recent advancements in computational power and techniques enable us to automate parts of the analyses that statisticians do—see for example the automatic statistician research project[4]. In our solution, we keep the implementation of the model open, that is, we allow using any model that can be applied to time series data.

As a use case, we selected the `auto.arima()` function provided in the `forecast` package in `R` [13]. The `auto.arima()` function fits a number of different ARIMA time series models with varying parameters and selects the one that yields the best tradeoff in accuracy and model complexity. Additionally, we implemented further naive time series predictors to compare prediction accuracies.

Avoiding Negative Values Imagine a negative trend in the durations of an activity—perhaps caused by a process participant getting more efficient in handling cases over time. If we simply extrapolate a negative trend when forecasting, we will eventually forecast negative duration values, which obviously make no sense. In our proposed solution, we limit the forecast durations and ratios to be positive (including zero) and replace negative forecast values with 0. Alternative solutions would be to use log-transforms data and predict in the log-space, and then to transform the predictions back by exponentiation. We did not use the latter approach, as it has problems with dealing with zero values. Zero values occur naturally in our setting, e.g., when a transition is not fired in a time unit of aggregation, the count value is 0.

Having introduced solution approaches to the challenges, let us discuss the most critical challenge for integrating time series approaches with business process models: the challenge of irregularly spaced data. There is an important trade-off, which we need

[3] The `tsclean()` method in the `forecast` package in `R` provides automatic interpolation of missing values and removal of outliers.

[4] The Automatic Statistician project: http://www.automaticstatistician.com

to keep in mind using the approach of aggregation. On the one hand, we gain efficiency, that is, we can reuse a forecast value for one hour for all the cases that need a forecast in that hour. On the other hand, we lose the patterns inside the unit of aggregation. Additionally, if we choose a too narrow time unit, we will end up with a lot of time units with missing values (time units, in which no single case was observed).

In the following, we shall evaluate the TSPN formalism with respect to its predictive performance in synthetic and real settings.

4 Evaluation

In the previous section, we presented a novel approach to capture temporal dependencies within business processes. To evaluate its usefulness in the business process domain, we have to evaluate the model's predictive performance with synthetic process models. In this way, we are able to identify possible conceptual advantages of the model given rather *clean* data. That is, we are interested in answering the question how much better a time series model would be in comparison to models that do not incorporate temporal dependency structures, in a setting with clear temporal dependency structures.

4.1 Experimental Setting

To conduct the experiment, we created TSPN models with 10 activities in sequence (we do not use complex control flow structures, because we want to isolate the prediction performance and not distort the results with synchronization effects). The activities have either a sinusoidal pattern (representing a seasonal pattern) of follow a random ARMA process. More specifically, the process parameters are randomly drawn according for the following processes.

The *sinusoidal* process uses the following equation given a time point t:

$$Y_t = \alpha + \gamma \cdot \sin(t \cdot \beta) + \epsilon_t \tag{1}$$

Here, α is the intercept or mean value, γ is the amplitude and β is the frequency. Additionally, the process has an attached normally distributed error term ϵ.

The *ARMA* process is generated by the following process:

$$Y_t = \sum_{i=1}^{n} \alpha_i y_{t-i} + \sum_{i=1}^{m} \beta_i \epsilon_{t-i} + \epsilon_t \tag{2}$$

This process consists of an autoregressive part with order n and parameters $\alpha_1, \ldots, \alpha_n$, and a moving average part with order m and parameters $\beta_1, \ldots, \beta_m$. It also includes a normally distributed error term ϵ.

We implemented the TSPN formalism in the open-source process mining software ProM[5]. The package is freely available as open-source software and the synthetic data sets are provided for testing as well. Figure 4 shows a screenshot of the plug-in visualizing an enriched sinusoidal TSPN process. The main window shows the process structure, and the lower windows project some statistical information about the durations. From

[5] See `StochasticPetriNet` package in ProM: http://www.promtools.org/

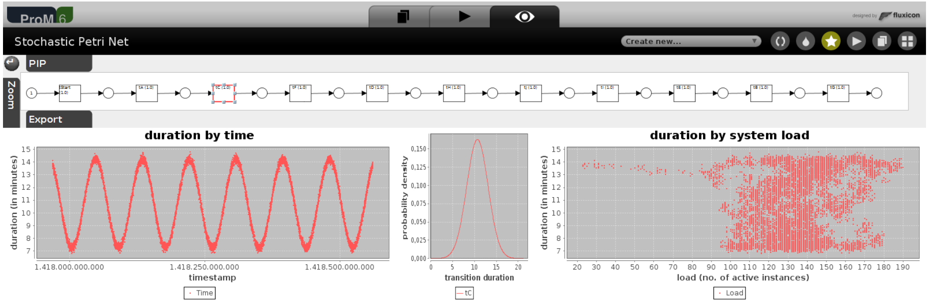


Figure 4: Screenshot of a sequential time series Petri net with a sinusoidal pattern of durations.

right to left this is the duration plotted as a scatter plot against the system load (i.e., the number of concurrently active cases), an aggregate plot of the collected duration values as a probability density function (middle), and the activity durations of the observed cases in relation to time (left). In this lower left screen, analysts can quickly identify temporal patterns in their process data, insofar as these are present.

After the creation of the synthetic models with their corresponding processes, we sample 10,000 process instances from each model and thereby obtain the simulated event logs. We will use these logs to test how different time series models and time agnostic models for prediction based on Petri nets [20] are able to capture the patterns in the processes.

For the evaluation, we need to ensure that our predictions are only based on available data. Notably, we cannot rely on a usual *cross-validation* approach when using time series data. Instead, we need a *rolling forecasting origin* approach. With a rolling forecasting origin, the first k observations $y_1, \ldots, y_k$ are used as training set to train the models that are then used to predict the next value with a forecast horizon h. This procedure is repeated with the next forecast using the next observations $y_{1+h}, \ldots, y_{k+h}$. In our case, we use 10 percent as training data and repeat this procedure for every further event in the event log.

Let us illustrate the approach with the example of predicting the duration of a single activity, which translates to predicting the firing time of the corresponding transition of the Petri net. In this case, the previous observations of that transition's duration are aggregated into a time series of the desired granularity—in our case into hourly averages. These aggregates are used as *training* data to fit the models that we want to compare.

Having fit the models, we then want to predict the duration of an activity. Therefore, we check the timestamp at the prediction time and compare it to the last observation's timestamp. The difference of the timestamps in hours is the forecast horizon h. For example, if we want to predict the duration of the service station for a customer that just called on Monday morning at 8am, and the last observation was on Friday 6pm, then the forecast horizon h is $6 + 24 + 24 + 8 = 62$ hours.

4.2 Compared models

As mentioned in the previous section, we selected the `auto.arima()` based model, which ideally selects the best fitting representative of a family of ARIMA models. But we also implemented four naive predictors common to time series analysis [12, Chapter 2.3]. Let us denote the number of observations in the time series as N, the predicted value of a model as $\hat{y}$, and the forecast horizon as h.

The *average method* completely ignores any temporal patterns. It predicts the next observation as the mean of the values observed so far and is defined as:

$$\hat{y}_{N+h|N} = \bar{y} = \frac{y_1 + \cdots + y_N}{N}. \tag{3}$$

The *naive method* uses the last observation as next predictor and ignores the horizon:

$$\hat{y}_{N+h|N} = y_N \tag{4}$$

The *seasonal naive method* (with a season-parameter m) is similar to the naive one, but it uses the observation from the last season to predict the duration:

$$\hat{y}_{N+h|N} = y_{N+h-km}, \quad \text{with } k = \left\lfloor \frac{h-1}{m} \right\rfloor + 1 \tag{5}$$

The *drift method*, allows the forecasts to linearly increase or decrease over time, where the amount of change over time (called the drift) is set as the average change of the historical data. So the forecast for time $t + h$ is given by:

$$\hat{y}_{N+h|N} = y_N + \frac{h}{N-1} \sum_{t=2}^{N} (y_t - y_{t-1}) \tag{6}$$

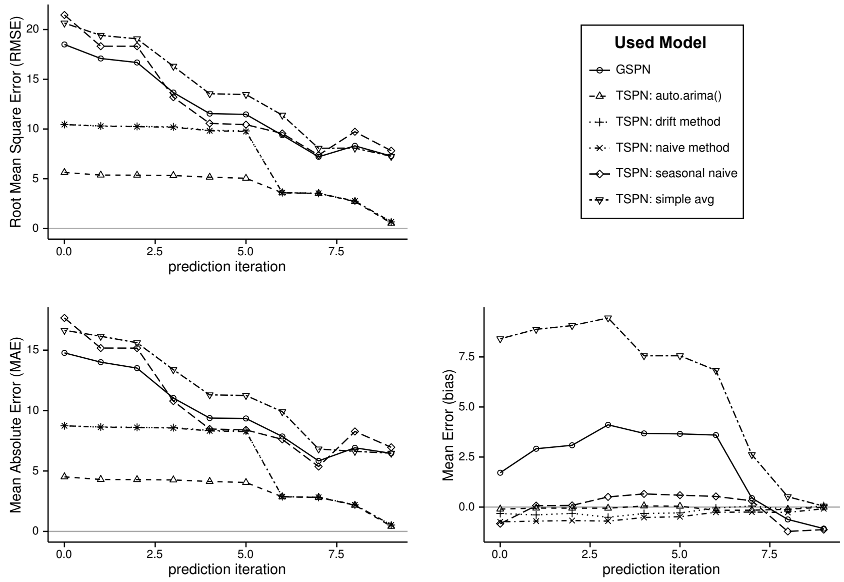
This is equivalent to drawing a line between the first and last observation, and extrapolating it into the future.

Further, we added two Petri net based time prediction models which do not consider seasonal or trend effects in the data. Namely, a GSPN model, and a generally distributed transition stochastic Petri net (GDT_SPN) model based on a non-parametric Gaussian kernel regression for the distribution of the duration observations. These models serve as a reference for models without time series features.

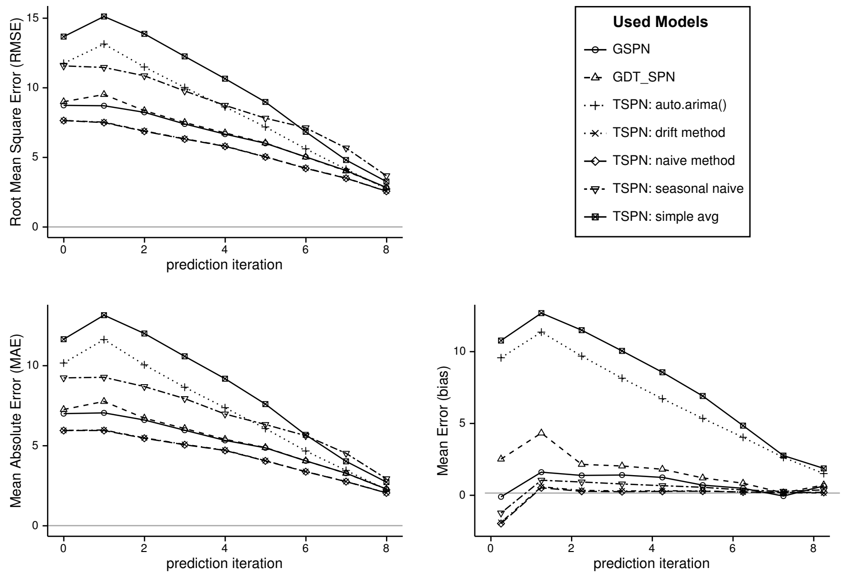
4.3 Evaluation Results

We are interested in how well the models can predict the observed behavior out of sample. For each prediction iteration (i.e., when a new event is observed) we compare the predicted remaining process duration to the actual remaining duration recorded in the log. The difference between the forecast an the actual value is called prediction *error*. By aggregating the errors in certain measures, we can weight the prediction quality of the different models against each other.

Therefore, we measure the *precision* of the model as the bias, which is represented by the mean error, and look at two *accuracy* measures: The mean absolute error (MAE) and the root mean square error (RMSE). Latter is more sensitive to our model predicting



(a) Results for the sinusoidal duration case.



(b) Results for the ARMA process case.

Figure 5: Prediction results for the sequential process with 5a sinusoidal duration patterns and 5b random ARMA processes generating transition durations. All competing models were run with the rolling forecasting origin method. The prediction is made at each new observed event based on an event log of 10,000 cases and a rolling forecasting window of 1,000 cases.

values far off the observed values, while the MAE is an easily interpretable measure: It tells us that on average, our model makes an error of the size of the MAE.

Figure 5 shows the prediction results of the competing models for the sinusoidal duration pattern Figure 5a and the ARMA-driven duration process Figure 5b. It can be read from the plots showing the RMSEs and MAEs that the `auto.arima()` based model fits the sinusoidal pattern well in comparison to the naive methods. In the ARMA case, the `auto.arima()` method does not fully capture the pattern of the underlying process. We also see that the two naive methods (the naive method using the last observation and the drift method that adjusts the last observation with a drift) are converging. This is expected, as the prediction horizon is mostly only one step, because there are no gaps in the data.

Note that the TSPN based approaches taking into account temporal relationships outperform those comparison methods in terms of accuracy of prediction that cannot make use of the temporal patterns in the data (i.e., the GSPN, GDT_SPN, and the simple average TSPN models).

With these promising results on the synthetic data sets, we now return to our illustrative example that we introduced in Section 2.1. We use the Petri net depicted in Figure 2 and a corresponding event log capturing 28 439 process instances of a call center process recorded in January 1999[6].

Figure 6 shows the prediction results for the call center case study. We conducted the same experiment as with the synthetic processes. Here, we observe that the differences of the various competing models are not substantial with the RMSE values of the predictors being in a similar range. This indicates, that this process does not have a temporal autocorrelation on hourly aggregate values that the compared forecast methods could exploit. However, it has to be noted that all time series predictors except for the `auto.arima()` perform equally well even without time series effects in the data.

4.4 Discussion of the Results

The results presented here can be summarized as follows. In the presence of time series effects as in the synthetic logs, our TSPN models can effectively exploit these for making better predictions than standard stochastic Petri nets. In the absence of time series effects as in the real-life log case study, our models perform equally well as the baseline. We observe that the `auto.arima()` function appears to be less robust in this case with partially creating biased estimates. Furthermore, it has to be noted that the temporal granularity might have an influence on the visibility of time series effects. For the case of the real-life data, we worked with hourly aggregates. At this stage, we cannot rule out that time series effects might be visible on a finer level. General guidelines towards the choice for a specific level of aggregation have to be inspected in future research.

5 Conclusion

With this paper we introduced the first model that integrates seasonal aspects and trends with the control flow structure of business process modeling. We provided the formal

[6] The data of the call center process is available at http://ie.technion.ac.il/Labs/Serveng

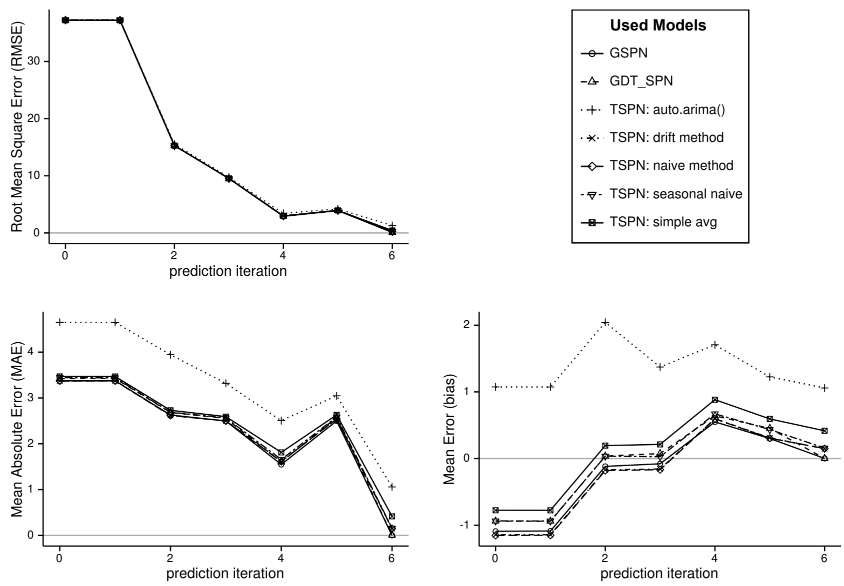


Figure 6: Prediction results for the call center case study. All competing models were run with the rolling forecasting origin method. The prediction is made at each new observed event for the model in Figure 2 and an event log containing 28,439 cases from January 1999.

model with its semantics, enrichment and an open-source implementation accompanied by the synthetic test data.

There remain some open research challenges to analyze in future work. For example, the results in the selected case study imply that there is potential for improvement on how to capture the exhibited patterns best. One branch of future research is to investigate more sophisticated methods in time series that are able to capture irregularly spaced time series data [7].

The potentials of time series methods are not yet fully unleashed with our approach, and more research is required in automatically selecting the granularity of the time series and the appropriate model type. Automatic elimination of holidays and integration of working schedules would further increase the accuracy of time predictions.

References

1. Wil M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.
2. Wil M.P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.
3. Wil M.P. van der Aalst, Michael Rosemann, and Marlon Dumas. Deadline-based escalation in process-aware information systems. *Decision Support Systems*, 43(2):492–511, 2007.

4. Wil M.P. van der Aalst, M. Helen Schonenberg, and Minseok Song. Time Prediction Based on Process Mining. *Information Systems*, 36(2):450–475, 2011.
5. George E.P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time series analysis: forecasting and control*. John Wiley & Sons, 4 edition, 2013.
6. Boudewijn F. van Dongen, Ronald A. Crooy, and Wil M.P. van der Aalst. Cycle time prediction: When will this case finally be finished? In *On the Move to Meaningful Internet Systems: OTM 2008*, volume 5331 of *LNCS*, pages 319–336. Springer, 2008.
7. Robert F. Engle and Jeffrey R. Russell. Autoregressive conditional duration: a new model for irregularly spaced transaction data. *Econometrica*, pages 1127–1162, 1998.
8. Francesco Folino, Massimo Guarascio, and Luigi Pontieri. Discovering Context-Aware Models for Predicting Business Process Performances. In *On the Move to Meaningful Internet Systems: OTM 2012*, pages 287–304. Springer, 2012.
9. Reinhard German. Non-Markovian Analysis. In *Lectures on Formal Methods and Performance Analysis*, volume 2090 of *LNCS*, pages 156–182. Springer, 2001.
10. Jan G. de Gooijer and Rob J. Hyndman. 25 Years of Time Series Forecasting. *International Journal of Forecasting*, 22(3):443–473, 2006.
11. David J. Hand, Heikki Mannila, and Padhraic Smyth. *Principles of data mining*. MIT press, 2001.
12. Rob J. Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2014. `https://www.otexts.org/book/fpp`.
13. Rob J. Hyndman and Yeasmin Khandakar. Automatic time series for forecasting: the forecast package for R. Technical report, Monash University, Department of Econometrics and Business Statistics, 2007.
14. Niels Lohmann, H.M.W. (Eric) Verbeek, and Remco Dijkman. Petri Net Transformations for Business Processes - A Survey. In *Transactions on Petri Nets and Other Models of Concurrency II*, volume 5460 of *LNCS*, pages 46–63. Springer, 2009.
15. Marco Ajmone Marsan. Stochastic Petri Nets: An Elementary Introduction. In *Advances in Petri Nets 1989*, pages 1–29. Springer, 1990.
16. Marco Ajmone Marsan, Gianfranco Balbo, Andrea Bobbio, Giovanni Chiola, Gianni Conte, and Aldo Cumani. The Effect of Execution Policies on the Semantics and Analysis of Stochastic Petri Nets. *IEEE Transactions on Software Engineering*, 15:832–846, 1989.
17. Marco Ajmone Marsan, Gianni Conte, and Gianfranco Balbo. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM TOCS*, 2(2):93–122, 1984.
18. Michael K. Molloy. *On the Integration of Delay and Throughput Measures in Distributed Processing Models*. PhD thesis, University of California, Los Angeles, 1981.
19. Johannes Prescher, Claudio Di Ciccio, and Jan Mendling. From declarative processes to imperative models. In *Proceedings of the 4th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2014)*, pages 162–173, 2014.
20. Andreas Rogge-Solti, Wil M.P. van der Aalst, and Mathias Weske. Discovering stochastic petri nets with arbitrary delay distributions from event logs. In *BPM Workshops*, volume 171 of *LNBIP*, pages 15–27. Springer, 2014.
21. Andreas Rogge-Solti and Gjergji Kasneci. Temporal anomaly detection in business processes. In *BPM*, volume 8659 of *LNCS*, pages 234–249. Springer, 2014.
22. Helen Schonenberg, Natalia Sidorova, Wil M.P. van der Aalst, and Kees van Hee. History-Dependent Stochastic Petri Nets. In *Perspectives of Systems Informatics*, volume 5947 of *LNCS*, pages 366–379. Springer, 2010.
23. Arik Senderovich, Matthias Weidlich, Avigdor Gal, and Avishai Mandelbaum. Queue mining – predicting delays in service processes. In *Advanced Information Systems Engineering*, volume 8484 of *LNCS*, pages 42–57. Springer, 2014.