Semi-Automated Time-Granularity Detection for Data-Driven Simulation Using Process Mining and System Dynamics

Mahsa Pourbafrani[1], Sebastiaan J. van Zelst[1,2], and Wil M. P. van der Aalst[1,2]

[1] Chair of Process and Data Science, RWTH Aachen University, Germany
{mahsa.bafrani,s.j.v.zelst,wvdaalst}@pads.rwth-aachen.de
[2] Fraunhofer Institute for Applied Information Technology (FIT), Germany
{sebastiaan.van.zelst,wil.van.der.aalst}@fit.fraunhofer.de

Abstract. Most information systems supporting operational processes also record event logs. These can be used to diagnose performance and compliance problems. The majority of process mining techniques extract models that are descriptive and describe what happened in the past. Few process mining techniques discover models that allow us to "look into the future" and perform predictive analyses. Recently, novel approaches have been developed for scenario-based prediction, i.e., predicting the effects of process changes on process performance, e.g., investing in an additional resource. To work accurately, the techniques need an appropriate time step-size, the selection of which, thus far, has been an ad-hoc and manual endeavor. Therefore, in this paper, building upon time-series analysis and forecasting techniques, we propose a novel semi-automated time-granularity detection framework. Our framework detects the best possible time-granularity to be used, whilst taking user preferences into account. Our evaluation, using both real and synthetic data, confirms the feasibility of our approach and highlights the importance of using accurate granularity in time step selection.

Keywords: Process mining · scenario-based predictions · system dynamics · what-if analysis · simulation · time-series analysis

1 Introduction

Process Mining [1] techniques derive knowledge of the execution of processes, by means of analyzing the data generated during their execution, which are stored in *event logs*. Several techniques exist, e.g., discovering a process model describing the process (*process discovery techniques* [7]), examining to what degree reality, captured in the data, conforms to a given process model (*conformance checking* techniques [10]), etc. Most techniques, extract models and insights that are descriptive. Few approaches focus on prescriptive/predictive models, i.e., models that allow us to "look into the future". Yet, at the same time, such techniques allow us to effectively *improve* the process, rather than just understanding its past performance.

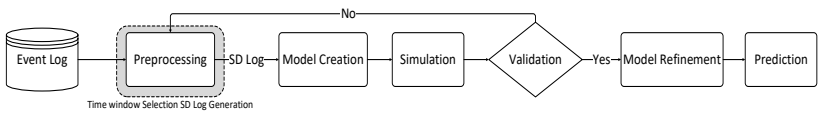

Fig. 1: The proposed framework for scenario-based process prediction, using system dynamics [16]. This paper focuses on *preprocessing* (highlighted), in particular, discovering the best time window for generating system dynamics logs.

In [16], we proposed a new process mining approach, i.e., *scenario-based prediction* of future process performance, using *System Dynamics* (SD) [21] as a prediction technique. The approach transforms an event log into a sequence of continuous variable values (e.g., process instance arrival rate), referred to as a *System Dynamics Log (SD-Log)*. The SD-Log forms the basis for simulation and prediction. Consider Fig. 1, in which we depict the general framework of the presented approach in [16]. First, we construct an SD-Log (Preprocessing step), and use it, together with a constructed model, to run a sample simulation ("as-is situation"). The quality of the simulations, i.e., both terms of validation and prediction, depends on the stability of the simulation model. In particular, the window-size of the time steps being used to generate the SD-Log highly affects the stability. Thus far, selecting such a window-size has been an ad-hoc/manual endeavor having negative effects on the prediction results.

Therefore, in this paper, we propose an approach that semi-automatically *identifies the best window-size* to be used in order to generate an SD-Log. Initially, the user provides a set of logical units (hours/days) based on domain knowledge that she/he wants to use in prediction. Subsequently, using the input event log, the proposed approach generates SD-Logs, based on several derivatives of the provided units. Subsequently, trend and pattern detection is applied to the different time-series, and correspondingly, the best step-size is selected. Within the trend and pattern detection, our approach is able to remove regular inactive time in the process. Using different real event logs, we assess the proposed approach including finding periodic behavior of the process including inactive steps. Then, we train different models and show the effect of the approach on reducing the prediction error. Furthermore, we use synthetic event logs with known patterns, including artificial noise/infrequent behavior, to test the feasibility of our approach. Our obtained results discover hidden patterns in the process variables and highlight the importance of selecting a suitable time step granularity.

The remainder of this paper is organized as follows. In Section 2, we present a running example. In Section 3, we introduce background concepts and notation. In Section 4, we present our main approach, which we evaluate in Section 5. In Section 6, we present related work. Section 7 concludes this work.

2 Running Example

In order to clearly demonstrate each step of our approach, we use a running example. We consider a simple, fictional process of a *car rental company*, i.e., called CARZ. Working days at CARZ are from Monday to Friday. The working hours are from 8:00 am to 5:00 pm (including 1 hour lunchtime). Requests for a rental car are received by phone. A different process is executed to handle the different types of requests, e.g., *rent a car* or *rent a car with a driver*. In the model, the time of the next call is derived from a normal distribution with 5 minutes average. The hours of the days affect the probability of generating new calls, e.g., the intensity of receiving calls at 10:00 am is 3 times higher than 8:00 am. If the number of callers in the queue is more than 20, new calls get rejected.

For each type of request, we use a (different) normal distribution to generate service times. The service time also gets affected by the number of requests in the queue. On average, the duration of handling *a car with a driver* request is 10 minutes higher than handling *rent a car* requests. We designed the model, such that operators perform the process of the calls faster if the number of calls in the line is higher. This effect, the queue length on time of processing calls, is modeled as an exponential nonlinear relation. We modeled the request handling process of CARZ, using CPN Tools [13].

3 Preliminaries

Here, we introduce background concepts and basic notation. We briefly cover common notions from the field of process mining, as well as system dynamics.

Process Mining Process mining techniques analyze the historical execution of processes, i.e., captured in the form of event logs, [1]. An event log captures what activity has been performed, at what time, for which instance of the process.

Definition 1 (Event Log). *Let ξ denote the universe of events. Furthermore, let $\mathcal{C}$, $\mathcal{A}$, $\mathcal{R}$ and $\mathcal{T}$ denote the universe of case identifiers, activities, resources, and the time universe, respectively. We define projections $\pi_{\mathcal{C}}{:}\,\xi \to \mathcal{C}$, $\pi_{\mathcal{A}}{:}\,\xi \to \mathcal{A}$, $\pi_{\mathcal{R}}{:}\,\xi \to \mathcal{R}$ and $\pi_{\mathcal{T}}{:}\,\xi \to \mathcal{T} \times \mathcal{T}$, s.t., given $e{\in}\xi$, we have $\pi_{\mathcal{C}}(e){=}c$, $\pi_{\mathcal{A}}(e){=}a$, $\pi_{\mathcal{R}}(e) = r$, and $\pi_{\mathcal{T}}(e){=}(t_s, t_c)$, indicating that event $e{\in}\xi$ captures the execution of an activity $a{\in}\mathcal{A}$, in the context of case $c{\in}\mathcal{C}$ by resource $r{\in}\mathcal{R}$, started at time $t_s{\in}\mathcal{T}$, and completed at time $t_c{\in}\mathcal{T}$. An event log L is a set of events, i.e., $L{\subseteq}\xi$.*

Table 1 depicts a snippet of a generated event log for the running example. The first row describes an event for which the activity *Process next Car Req* is executed by *Monika* for a request with *case ID* 10. An event log may include more data attributes, e.g., here type of requests is also logged (`CarRequest` or `DriverRequest`), but, for simplicity, we abstract from such additional attributes.

Table 1: Sample event log, generated for the CARZ running example. Each row is an event in which for each unique customer (case) in the process, a specific activity at a specific time is performed by a specific resource.

Case ID	Activity	Request Type	Timestamp	Complete Timestamp	Resource
10	Next Call	CarRequest	1/1/2018 10:29	1/1/2018 10:47	Monika
11	Next Call	DriverRequest	1/1/2018 10:29	1/1/2018 10:29	System
8	Process Next Driver Req	DriverRequest	1/1/2018 10:30	1/1/2018 10:50	Pheobi
10	Process Next CarReq	CarRequest	1/1/2018 10:31	1/1/2018 10:49	Chandler
13	Next Call	DriverRequest	1/1/2018 10:31	1/1/2018 10:31	System
10	Processed CarReq	CarRequest	1/1/2018 10:32	1/1/2018 10:32	System
⋮	⋮	⋮	⋮	⋮	⋮

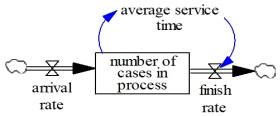

Fig. 2: Simple stock-flow diagram. The value of the stock *number of cases in process* is calculated based on the *arrival rate* and *finish rate* flows (per time step). The value of *finish rate* is affected by the *average service time*.

System Dynamics System dynamics techniques are used to model dynamic systems and their relations with their environment [21]. One of the main modeling notations in system dynamics is the stock-flow diagram, which models the system w.r.t. three different elements, i.e., stocks, flows and variables [19]. Stocks are accumulative variables over time, flows manipulate the stock values and variables influence the values of flows and other variables over time. Figure 2 shows a simple stock-flow diagram for the example in which *arrival rate* and *finish rate* as flows add/remove to/from the values of *number of cases in the process* as stock, also, *average service time* as a variable affects the finish rate based on the number of cases in the process.

System Dynamics Logs Event logs do not suffice to populate a given system dynamics model with values for stocks, flows, and variables, therefore, they should be transformed into an actionable form, i.e., numerical values. Hence, we define the notion of a *System Dynamics Log (SD-Log)*, i.e., a sequence of continuous variable values, capturing the numerical values for a set of variables of interest over time, as described by the event log. Assume that, the first event in an event log starts at time t_s, and, the last event is completed at time t_C. Given time window $\delta \in \mathbb{N}_{\geq 0}$, there are $k = \lceil (t_C - t_S)/\delta \rceil$ subsequent time steps in the event log for time window δ. An SD-Log captures all the values for the variables of interest, in each time-window.

Definition 2 (SD-Log). *Let $L \subseteq \xi$ be an event log, let $\mathcal{V}$ be a set of process variables, and let $\delta \in \mathbb{N}_{\geq 0}$ be the selected time window. Let t_S denote the minimal start timestamp in L, let t_C denote the maximal end timestamp in L and let $k = \lceil (t_C - t_S)/\delta \rceil$. An SD-Log of L, given δ, $sd_{L,\delta}$, is a multivariate time-series, i.e., $sd_{L,\delta} \in \{1, ..., k\} \times \mathcal{V} \rightarrow \mathbb{R}$, s.t., $sd_{L,\delta}(i, v)$ represents the value of process variable $v \in \mathcal{V}$ in the i^{th}-time window $(1 \leq i \leq k)$.*

Given an event log L, a set of variables $\mathcal{V}$, and window δ, the event log is transformed log into an SD-Log. If L is clear from the context, we omit it and write

Table 2: Example derived SD-Log for the running example with a time window of 1 day and 6 different process variables. Each row shows a time step, here 1 day, cell-values represented aggregated variable values.

Time Window Daily	Arrival rate	Finish rate	Num of unique resources	Avg service time	Avg time in process	Avg waiting time in process
1	180	180	6	0.3590	0.9689	0.6099
2	147	147	6	0.4156	0.9565	0.5409
3	160	160	6	0.4011	0.9972	0.5961
4	116	116	6	0.4455	0.9363	0.4908
5	94	94	6	0.5024	0.8258	0.3234
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	147	147	6	0.4421	0.9898	0.5477
⋮	⋮	⋮	⋮	⋮	⋮	⋮

sd_δ. Given sd_δ and $v \in \mathcal{V}$, we write $\Pi_v(sd_\delta) \in \mathbb{R}^*$, returning the sequence of values $\langle x_1, ..., x_k \rangle$ for variable v. Furthermore, π_i returns the i^{th} value in a sequence, for instance, $\pi_i(\Pi_v(sd_\delta)) = x_i$. In the running example, consider the set of variables $\mathcal{V} = \{arrival\ rate, average\ service\ time, number\ of\ people\ in\ the\ process\}$, for a duration of 14 days with $\delta = 1\ day$, i.e., the corresponding SD-Log includes 14 time steps. Consider Table 2, in the first time window (day) 180 cases were arrived at the process and 6 unique resources were performing the tasks. $\Pi_v(sd_\delta) = \langle x_1, ..., x_k \rangle$ is a series of values over steps of time with length k, which is in the form of time-series data.

Time-Series The analysis of sequences of real values and/or sequences of tuples of real values is often referred to as *time-series analysis* [12]. Several models exist that, given a sequence of values and/or tuples of values, predict the next (sequence of) number(s). Examples include *Moving Average models* (MA), *Auto-Regressive models* (AR), and *Auto Regressive Integrated Moving Average models* (ARIMA). The exact type of model used to predict the next likely values is not relevant for our approach, i.e., any method that allows us to do so suffices. Hence, in Definition 3, we propose a generic definition of a *time-series model*.

Definition 3 (Time-Series Model). *Let $\sigma = \langle x_1, ..., x_k \rangle \in \mathbb{R}^*$ be a sequence of real values (a time-series). A time-series model θ is a function $\theta: \mathbb{R}^* \to \mathbb{R}^*$. Given $\sigma = \langle x_1, ..., x_k \rangle$, $\theta(\sigma) = \langle \hat{x}_1, ..., \hat{x}_k \rangle$, s.t., for $1 \leq i \leq k$: $\hat{x}_i$ is the expected value for x_i.*

Observe that Definition 3 covers *univariate time-series*. For predicting the first value (x_1), we use random initial values. To measure the accuracy of the time-series model (θ), we use *Mean Absolute Percentage Error* ($MAPE = \frac{100\%}{k} \sum_{i=1}^{k} |\frac{x_i - \hat{x}_i}{x_i}|$).

4 Proposed Approach

Consider Fig. 3, in which we depict an overview of the approach. The approach starts with an event log and logical units of time, as shown in the top left side of Fig. 3. The logical units can be minutes, hours, days, etc. Furthermore, units

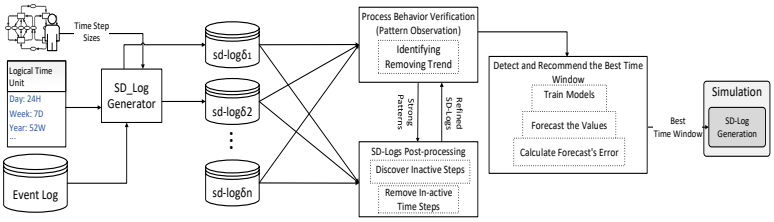

Fig. 3: Proposed approach for discovering the best time window, generating/analyzing time-series data from event logs, investigating the effect of business processes inactivity and detecting strong patterns in the processes over time.

are related to one-an-other, e.g., days consist of 24 hours, weeks are 7 days, etc. Our approach starts with a set of initial sizes of time steps, Δ, provided by the user. Given a set of process variables $\mathcal{V}$ and the set of different sizes Δ, for each $\delta \in \Delta$ by the user, a corresponding SD-Log sd_δ is calculated. The derived values in the SD-Logs are tested for repetitive patterns over time, i.e., regular behavior (*Process Behavior Verification* step). Inactive steps are removed using the discovered regular inactive patterns (*SD-Logs Post processing*). If the SD-Log shows patterns of inactivity, then all the corresponding inactive steps are removed. The last step is to find the best time window for extracting the values for simulation models by training time-series models as explained in Section 4.3.

4.1 Process Behavior Observation

Observing the process behavior over time makes it possible to see and discover periodic patterns. We define function *Test Time Step (TTS)* to discover strong patterns for process variables that show repetitive behavior in the context of the process environment over time, e.g., *arrival rate*. We use the partial auto-correlation function [20] to find the possible existing patterns in $\Pi_v(sd_\delta)$ for variable $v \in \mathcal{V}$, for each derived sd_δ, $\delta \in \Delta$, where Δ is provided by the user. In real event logs, process variables over time, e.g., arrival rate, can be highly correlated to the previous values of themselves, hence, computing the partial auto-correlation allows us to remove such internal dependencies. By doing so, we only consider the correlation between two lagged-values, aim in finding clear patterns inside the data. The *lag-value* shows that the correlation between which pair of values should be calculated.

Definition 4 (Test Time Step). *Let $\sigma \in \mathbb{R}^*$ and $T \subseteq \mathbb{N}$ be the set of possible lag-values. $PAC_\tau : \mathbb{R}^* \rightarrow [-1, 1]$ defines the partial auto-correlation of given lag-value $\tau \in T$. Function $TestTimeStep$ is defined as $TTS_\rho : \mathbb{R}^* \rightarrow 2^T$, where $\rho \in \mathbb{R}_{\geq 0}$. For $\sigma \in \mathbb{R}^*$ and threshold ρ, $TTS_\rho(\sigma) = \{\tau \in T | abs(PAC_\tau(\sigma)) \geq \rho\}$.*

By definition, the value of the partial auto-correlation function is always 1 for lag 0. Figure 4 shows the partial auto-correlation values as a sample for the process arrival rate in an hourly and daily manner. For the running example,

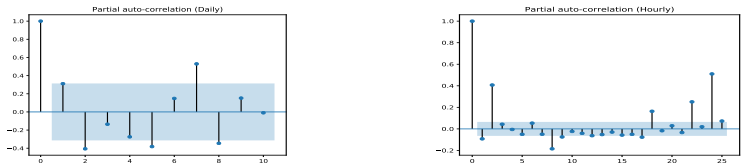

Fig. 4: The partial auto-correlations for the process arrival rate. Daily (left) and hourly (right) time windows (left).

consider sd_{hour} as the derived SD-Log and *arrival rate* as a process variable $v{\in}\mathcal{V}$, $\Pi_{arrival\ rate}(sd_{hour})$ returns a sequence of values for arrival rate per hour. For ρ=0.5, the function TTS_ρ over the derived sequence, returns $\{24\}$, i.e., the process shows similar/stable behavior every 24 hours.

4.2 SD-Log Post-Processing

In addition to the patterns inside the process variables for different sizes of time steps, the *inactivity* of the process in each step is also important. There are time steps in which the process is inactive. Such inactivity can either be planned/intentional or, unexpected. Differentiating different types of inactivity is required in order to capture the most stable behavior of the process. This behavior is directly affecting the simulation results. In this step, first, we need to discover the inactive steps in the process and then, using the previous step, TTS function result, remove periodic and regular inactive steps. Function *Detect Inactivity* (Definition 5) discovers the inactive steps of time for the process. The function maps each step of time in the SD-Log into a boolean value, indicating whether or not there are reasons to believe that the process was inactive, in that time step. Inactivity is measured on the basis of all the process variables $\mathcal{V}$ combined, i.e., there has to be a significant amount of variables that show inactivity to classify the step as an inactive step.

Definition 5 (Detect Inactivity). *Let $\mathcal{V}$ be the set of process variables and $|\mathcal{V}|= n$, let $\gamma{\in}\mathbb{R}_{>0}$, and let $\kappa{\in}\mathbb{R}^n$ denote a vector of thresholds for considering a variable as active. $DIA_\kappa{:}\mathbb{R}^n{\to}\{0,1\}$ is a function describing the relative inactivity of a given $\mathbf{x}{\in}\mathbb{R}^n$, subject to activity threshold κ, i.e.,:*

$$DIA_{\kappa,\gamma}(\mathbf{x}) = \begin{cases} 0 & \textit{if } \frac{|\{i{\in}\{1,\ldots,n\}|\mathbf{x}(i){\geq}\kappa(i)\}|}{n} \geq \gamma \\ 1 & \textit{if } \frac{|\{i{\in}\{1,\ldots,n\}|\mathbf{x}(i){\geq}\kappa(i)\}|}{n} < \gamma \end{cases}$$

For instance, given SD-Log sd_δ and the set of variables $\mathcal{V}$, function $DIA_{\kappa,\gamma}$ returns 0 if the relative number of values in each time step is above γ, otherwise it returns 1. The function indicates whether a time step in the process is active or inactive. The output of the *DIA* function, is used as input of *TTS* function. By applying *TTS* function on the result of the *DIA* which is a sequence over time, we discover whether there are any strong patterns inside the inactive steps.

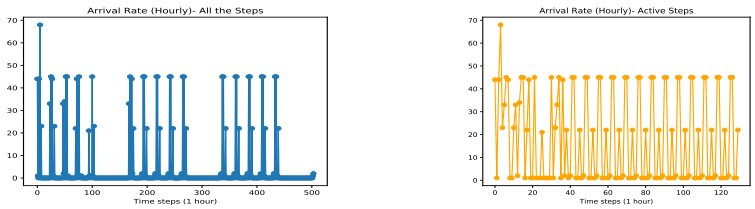

Fig. 5: The arrival rate of the running example in 1 hour steps for 3 weeks before (left) and after (right) removing regular inactivity. In the active steps, the minimum values are 1. The weekends and night hours have been removed.

The strong patterns reveal the periodic inactivity, then we remove the inactive steps from the SD-Log. We call the new refined SD-Log sd'_{δ}. The option for users with domain knowledge is available here to either remove or keep the detected inactive time steps. In the running example, the result of *Detect inactivity function DIA* for all the steps is $\langle 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0 ... \rangle$, where 0-values are weekends. Applying the *TTS* function, again returns 7 days as a strong pattern in the data, hence, the inactive time steps, i.e., weekends, are removed. Figure. 5 shows the hourly arrival rate of the process for 3 weeks before and after removing regular inactivity. The removed hours are nights and weekends. Note that in Fig. 5, the minimum value in the active steps is 1. Furthermore, applying *TTS* after removing the regular inactive steps, make it possible to discover whether there are interesting patterns inside the active steps of the process.

4.3 Detect the Best Time Window

After pattern detection and removing inactive time steps, we aim to find the best window of time, in order to generate the SD-Logs and perform simulations.

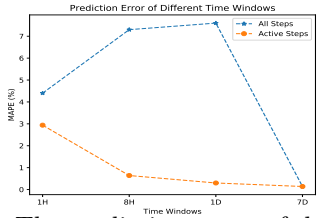

Fig. 6: The prediction errors of the models (ARIMA [9]) for different time windows. The models are trained based on the values of arrival rate, before (blue) and after (red) applying the proposed approach on the SD-Logs.

Using *TTS* and *DIA* functions, we transform the values of process variables in the SD-Logs into the steady time series values, which the frequent patterns and inactivity have been removed. We are looking for the time step size that displays the most stable behavior of the process. We use a time-series prediction model θ, to predict the expected values of a process variable (Definition 3). By *differencing* the values of the generated time-series data for the process variables, i.e., computing the differences between consecutive observations, the data becomes stationary and can be

used in time-series models, e.g., *ARIMA*. The accuracy of the models, for each of the selected time windows, shows the best time window to be used in prediction. For each time window $\delta \in \Delta$ provided by the user and corresponding SD-Log after removing inactivity sd'_δ, the values for v is $\Pi_v(sd'_\delta)=\langle x_1, ...x_k\rangle$. After training the models, we calculate the *Mean Absolute Percentage Error* (MAPE). Among all the tested sizes of the time step, the one with the minimal MAPE value indicates the best time window to be used for extracting SD-Log and performing the simulation. Variable *arrival rate* often is the only variable that shows the influence of the environment directly on the process. However, in the case of more variables, the average MAPE value is considered.

In the example, if $\Delta = \{hour, 8hours, day, 7days\}$, the SD-Logs are sd_{hour}, sd_{8hours}, sd_{day}, and sd_{7days}. For the process variable $v = arrival\ rate$, TTS and DIA are performed on the results of the projection function Π_v. In this step, ARIMA models are trained with different parameters and the prediction error for each δ is shown in Fig. 6 (red). The errors indicate that for the time window of 1 day or 7 days the values of arrival rate are more stable. Figure 6 also demonstrates the errors of the trained models before applying the steps of our approach (blue). Since there is no inactive week, the error for 7 days is the same, however, for other steps, after removing the inactivity, the error has reduced, e.g., removing weekends from daily time step, resulted in more stable and predictable behavior in the process.

5 Evaluation

To evaluate the approach, we use both synthetic and real event logs. Using the synthetic event log, we assess the effect of choosing time windows on the simulation accuracy. We discover the strong patterns in the real event logs.

Implementation The experiments are conducted with an implementation of the framework in the *PMSD* tool [15]. For time-series prediction, used in best time window detection (Section 4.3), we use ARIMA [9]. To compute ARIMA models, we need to set three parameters, i.e., differencing-parameter d, AR-term $p \geq 1$ and MA-term $q \geq 1$. In the remainder, we write $ARIMA(p, d, q)$.

5.1 Synthetic Event Log: Simulation Case Study

In order to assess the effect of the selected time window on the simulation results, we use the results of the performed steps on the running example through the sections to generate a system dynamic simulation model. Consider that businesses might be interested in a smaller window of time for prediction, e.g., a daily manner is more useful for decreasing the average daily waiting time rather than a weekly manner in the process. Therefore, the time window with the minimum error is not always the best option for the businesses. We generate the system dynamic models using the technique presented in [17]. The designed model in Fig. 7 is populated with two different SD-Logs, sd_{day} and sd_{hour}, after applying

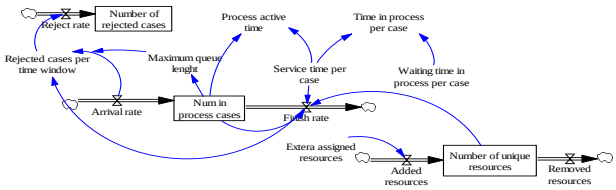

Fig. 7: System dynamics model for the running example. What-if analysis for the number of rejected cases per time window and the number of resources.

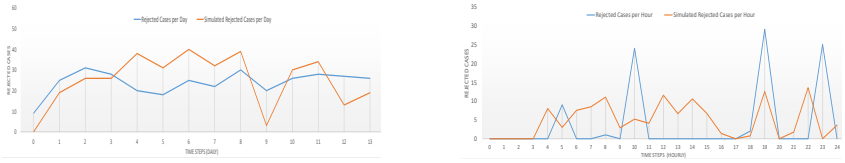

Fig. 8: The actual (blue) and simulated (red) number of rejected cases using daily (left) and hourly (right) time windows using the model in Fig. 7.

the approach. The target scenario is to simulate the number of rejected people in the process per time window and the effect of the number of resources to decrease the rejected cases.

Figure. 8 shows the results of the simulation for two selected time windows and the actual values from the event logs. The simulation results using the 1 day time window is close to reality behavior over 1 hour time window. As expected from the results of the approach, the time windows with lower errors provide more accurate simulation results. Note that, we designed the process in Section 2 such that the number of requests is higher for the specific time of the days, e.g., at 10:00 am the number of requests is more than 8:00 am. Therefore, even after removing regular inactive hours from sd_{hour}, the variation between the values in high. The results illustrate that selecting a proper time window to extract the values of variables highly affects the accuracy of the predictions and our approach is able to provide the information needed to pick the best time window.

5.2 Feasibility Test on Real Event Data

We use two real event logs, BPI Challenge 2012 [24] and BPI Challenge 2017 [25], to evaluate our work. The errors of the models for predicting the values of variables in different time windows, before and after performing the steps of the approach show the effect of the approach on selecting the best time window.

BPI Challenge 2012 We start with four time window sizes, *2 hours, 8 hours, 1 day and 7 days* and extract the SD-Logs for each time window. Using the function *TTS* in the *process behavior observation* step, the strong patterns in the values of arrival rate are discovered, e.g., with threshold 0.5, there are strong

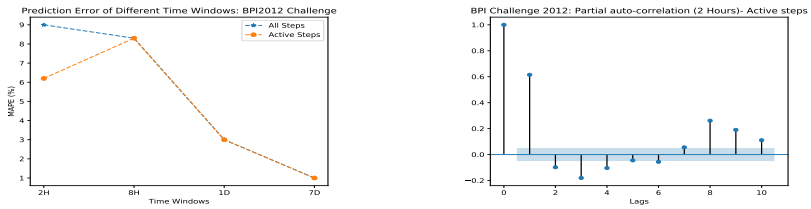

Fig. 9: The prediction errors of the trained models for time windows before and after removing inactive steps, BPI 2012 (left). The partial auto-correlation after removing inactive periods for 2 hour time window (right).

patterns in every 2 hours (lag 1 for 2 hours time window in Fig. 9 (right)) and 7 days for the daily time window. *Post processing* step and DIA function result in the refined SD-Logs for the selected time windows by removing possible periodic inactive steps. In Fig. 9 (left), we present the error of the training ARIMA model with different parameters before (blue) and after (red) removing inactive steps. For instance, the best one for the hourly window after removing the inactive steps is $ARIMA(2,0,1)$ and for 1 day window is $ARIMA(1,0,1)$. Since in the process, there is no inactive week, day and 8 hours, therefore the prediction errors are the same for including all the steps and active steps. However, the error of the 2 hours time window has decreased. The reduction shows that in the process there were periodically inactive steps between each 2 hours. As expected, the strongest pattern inside the process w.r.t. its environment, i.e., the arrival rate of cases, is *7 days*. Furthermore, the time window sizes based on the domain knowledge can be changed, e.g., 8 hours is tested to see whether the process follows the common working hours.

BPI Challenge 2017 In the process, there are different types of activities. We focus on activities which are triggered by the employees, (activities with a $W\_$ prefix). We also use *8 hours*, *1 day* and *7 days* as time windows with respect to the employees' working hours. Figure 10 represents the partial auto-correlation of the arrival rate (daily and 8 hours) in *behavior observation* step. As expected, the strongest pattern is every 7 days (weekly). Also, in 8 hours time window, lag 3 shows a strong pattern.

DIA function is applied on the SD-Logs to indicate the inactive steps and using results of TTS to remove the regular inactive steps, the process shows more stable behavior. For instance, Fig. 11 (right) is the TTS result on the daily arrival rate after removing weekends, hence there is no more strong pattern. This information helps in analyzing the process behavior and have more accurate simulation models. The prediction errors of the trained models for predicting the values of the arrival rate for three selected time windows are presented in Fig. 11 for the derived SD-Logs and the refined SD-Logs using our approach. Same as the BPI Challenge 2012, there are no regular inactive weeks in the process, therefore the error has not changed before and after applying the approach. However, in both 8 hours and 1 day time windows, there are considerable reductions in

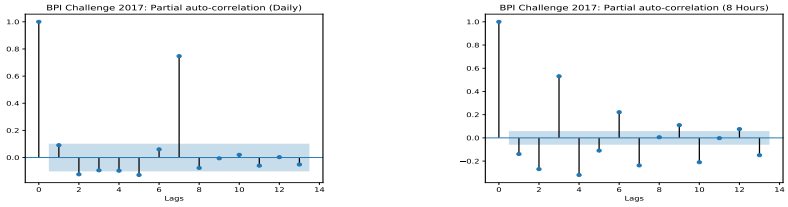

Fig. 10: The partial auto-correlation before removing inactivity in 1 day window (left) and 8 hours time window (right) for the BPI challenge 2017.

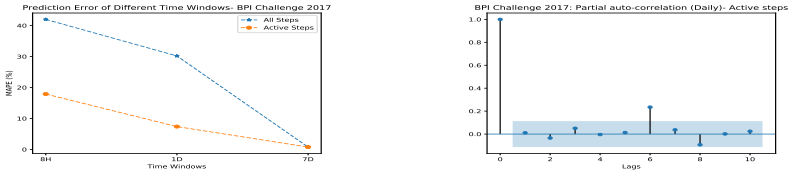

Fig. 11: The prediction errors of the arrival rate in the BPI 2017 event log before and after removing the inactive time steps (left). The partial auto-correlation after removing inactive time steps (right).

the prediction errors. The evaluation using real event logs indicates that the approach is able to find a better time window among the possible time step sizes to have the most stable behavior of the processes.

6 Related Work

Process mining techniques implicitly use time-series data for different purposes such as performance analysis [3], bottleneck analysis, prediction [2] and the enhancement of the processes, e.g., providing recommendations [5]. Process mining techniques mostly focus on the current state of processes. At the same time, in simulation, the current state is employed to generate similar behavior of the processes. Therefore, the combination of these two fields is a promising direction for the enhancement of processes [2].

Prediction and simulation techniques in process mining focus mostly on instances of the process, e.g., the execution/ waiting time for a specific case in the process [22]. Moreover, in [6] a configurable approach is proposed to construct a process model enhanced with time information from previous instances, and use the model for prediction, for example, the completion time. However, most of the mentioned techniques are at a detailed level and missing the effect of external factors [4]. Furthermore, in [23] a survey of prediction techniques in process mining is presented in which most of the techniques use a predefined unit of time such as days or hour. In the context of simulation and prediction in process mining, there is not enough focus on the effect of the size of time windows on the result of simulation and prediction techniques. Work such as [11] explains

the possibility of using time-series analysis in data analysis. Two main types of time-series analysis exist including univariate and multi-variate. Box et al. introduced the ARIMA method [8]. This method now represents one of the most frequently used univariate time-series modeling tools. In [14], techniques such as the ARIMA technique are shown to be more effective than LSTM techniques in univariate time-series data [22].

In most of the techniques in process mining, the selection of the time window for generating data either has not been mentioned explicitly or they used the predefined logical unit of time. The techniques at the aggregated level use the current state of the processes over time. Techniques such as [16] are proposed which employ different time windows. This approach can be used with domain knowledge about the working hours of the processes, e.g., a production line process is running 8 hours per day [18].

7 Conclusion

In this paper, we proposed an approach to discover the best window of time for capturing the most stable behavior of processes over time. The discovered time window is used for extracting values of process variables from event logs. Since these values are an aggregated value for each time window, they behave like time-series data. We used the derived time-series data to discover strong patterns inside the process variables related to the process environment, e.g., arrival rate. Moreover, in the approach, inactive time windows are distinguished and removed. A time-series prediction approach (ARIMA) is used to find the best models that predict the next values accurately and their parameters.The proposed approach is effective in picking the size of the time window to generate the performance variables in business processes. The generated values for process variables over time that represent the process behavior, i.e., SD-Log, are exploited for simulation and prediction purpose. The evaluation section shows that our approach provides business owners with actionable insights into the current situation of the processes to be used in simulation as system dynamics.

Acknowledgments Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy–EXC-2023 Internet of Production – 390621612. We also thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016). https://doi.org/10.1007/978-3-662-49851-4
2. van der Aalst, W.M.P.: Process Mining and Simulation: A Match Made in Heaven! In: D'Ambrogio, A., Zacharewicz, G. (eds.) Computer Simulation Conference (SummerSim 2018). pp. 1–12. ACM Press (2018)

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 Interdiscip. Rev. Data Min. Knowl. Discov. **2**(2), 182–192 (2012)
4. van der Aalst, W.M.P., Dustdar, S.: Process mining put into context. IEEE Internet Computing **16**, 82–86 (2012)
5. van der Aalst, W.M.P., Pesic, M., Song, M.: Beyond process mining: From the past to present and future. In: CAiSE 2010. pp. 38–52 (2010)
6. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Inf. Syst. **36**(2), 450–475 (2011)
7. Augusto, A., Conforti, R., Dumas, M., Rosa, M.L., Maggi, F.M., Marrella, A., Mecella, M., Soo, A.: Automated discovery of process models from event logs: Review and benchmark. IEEE Trans. Knowl. Data Eng. **31**(4), 686–705 (2019)
8. Box, G.E.P., Jenkins, G.: Time Series Analysis, Forecasting and Control. Holden-Day, Inc., USA (1990)
9. Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time series analysis: forecasting and control. John Wiley & Sons (2015)
10. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018)
11. Esling, P., Agón, C.: Time-series data mining. ACM Comput. Surv. **45**(1), 12:1–12:34 (2012). https://doi.org/10.1145/2379776.2379788
12. Hamilton, J.D.: Time series analysis, vol. 2. Princeton New Jersey (1994)
13. Jensen, K., Kristensen, L.M.: Coloured Petri Nets - Modelling and Validation of Concurrent Systems. Springer (2009). https://doi.org/10.1007/b95112
14. Makridakis, S., Spiliotis, E., Assimakopoulos, V.: Statistical and machine learning forecasting methods: Concerns and ways forward. PloS one **13**(3) (2018)
15. Pourbafrani, M., van der Aalst, W.M.P.: PMSD: Data-driven simulation in process mining. In: BPM 2020 (2020)
16. Pourbafrani, M., van Zelst, S.J., van der Aalst, W.M.P.: Scenario-based prediction of business processes using system dynamics. In: OTM 2019 Conferences, 2019. pp. 422–439 (2019). https://doi.org/10.1007/978-3-030-33246-4_27
17. Pourbafrani, M., van Zelst, S.J., van der Aalst, W.M.P.: Supporting automatic system dynamics model generation for simulation in the context of process mining. In: BIS 2020, Colorado Springs, CO, USA, June 8-10, 2020, Proceedings. pp. 249–263 (2020). https://doi.org/10.1007/978-3-030-53337-3_19
18. Pourbafrani, M., van Zelst, S.J., van der Aalst, W.M.P.: Supporting decisions in production line processes by combining process mining and system dynamics. In: Intelligent Human Systems Integration 2020 - Proceedings of the 3rd International Conference on Intelligent Human Systems Integration. pp. 461–467 (2020). https://doi.org/10.1007/978-3-030-39512-4_72
19. Pruyt, E.: Small System Dynamics Models for Big Issues: Triple Jump towards Real-World Dynamic Complexity. TU Delft Library (01 2013)
20. Ramsey, F.L., et al.: Characterization of the partial autocorrelation function. The Annals of Statistics **2**(6), 1296–1301 (1974)
21. Sterman, J.: System Dynamics: Systems Thinking and Modeling for a Complex World (2002)
22. Tax, N., Verenich, I., Rosa, M.L., Dumas, M.: Predictive business process monitoring with LSTM neural networks. CoRR **abs/1612.02130** (2016)
23. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. TKDD **13**(2), 17:1–17:57 (2019)
24. van Dongen, B.F.: BPIC 2012. Eindhoven University of Technology (2012)
25. van Dongen, B.F.: BPIC 2017. Eindhoven University of Technology (2017)