Business Trend Analysis by Simulation

Helen Schonenberg, Jingxian Jian, Natalia Sidorova, and Wil van der Aalst

Eindhoven University of Technology,
Department of Mathematics & Computer Science,
Den Dolech 2, 5600 MB Eindhoven, The Netherlands
{m.h.schonenberg, n.sidorova, w.m.p.v.d.aalst}@tue.nl

Abstract. Business processes are constantly affected by the environment in which they execute. The environment can change due to seasonal and financial trends. For organisations it is crucial to understand their processes and to be able to estimate the effects of these trends on the processes. Business process simulation is a way to investigate the performance of a business process and to analyse the process response to injected trends. However, existing simulation approaches assume a steady state situation. Until now correlations and dependencies in the process have not been considered in simulation models, which can lead to wrong estimations of the performance. In this work we define an adaptive simulation model with a history-dependent mechanism that can be used to propagate changes in the environment through the model. In addition we focus on the detection of dependencies in the process based on the executions of the past. We demonstrate the application of adaptive simulation models by means of an experiment.

1 Introduction

Business processes are often the result of a fit between the needs and capabilities of the internal stakeholders of the business and the opportunities and threats the business identifies in its environment [14]. The environment in which these processes operate is typically unstable and business processes should be robust enough to cope with a variable and changing environment. The behaviour of the environment can be subject to seasonal or financial trends, such as customers not booking expensive holidays due to financial crisis. The most interesting question from the business point of view is: *"How will these trends affect the performance of my business process?"*.

This paper aims at detecting dependencies in the business process that can be used to accurately analyse the effects of environmental trends on the business process performance. Nowadays, most business processes are supported by information systems that store information about the process execution in logs. We can use this historical information to estimate the effect of trends on business process performance and to help organisations with obtaining insight in questions such as: *"Do we have enough resources available to execute the process during the holiday season?"*.

Real-life business processes usually contain many execution alternatives due to choices, parallelism, iterations and (data) dependencies in the process. For example the choice the environment makes in some part of the process might be correlated with a choice that the environment made earlier in the process, i.e. a customer who books an expensive mean of transportation is more likely to book more expensive hotels. Performance analysis of such systems by using analytical models is often intractable and simulation is used instead. For accurate simulation, i.e. simulation that is close to reality, it is of crucial importance to capture the real behaviour of the process. It is not sufficient to only include the actual execution alternatives in the simulation model to analyse performance, also information about decisions, costs, resources and stochastic aspects of the behaviour need to be included [17].

In most simulation tools for business process management, simulation parameters, like activity cost, duration and probability, are variables that are assumed to be independent, which is often not the case. Incorrect assumptions about correlations and dependencies can lead to over or underestimation of the outcome [9, 12, 18]. In this paper we show how dependencies are mined from a log of a business process and how they are incorporated into the simulation model. Our approach is to create an adaptive simulation model with parameters that adapt according to the information obtained by the simulation steps executed so far (history-dependency), based on the dependencies found in the log.

An adaptive simulation model is created from two components. The first component is the control flow model that can be either given (predefined models) or mined from the log using standard process mining techniques [2]. The second component consists of information about the simulation parameters. Again, the parameters can be predefined or the log can be used to determine the parameters. In this paper we focus on the latter case, where we consider the simulation parameters as random variables that we are going to estimate on the log. Both components are integrated into an adaptive simulation model by a history-dependent mechanism, that, for each instance, estimates the parameters on the (partial) simulation trace of that instance.

The outline of the paper is as follows. First we give some preliminaries in Section 2. In Section 3 we describe the adaptive simulation model. In practice abstractions will be needed to detect dependencies and Section 4 gives an overview of some elementary abstractions and shows how they can be combined. Section 5 illustrates the use of adaptive simulation models in the experimental setting. Related work will be presented in Section 6. Finally, we conclude the paper in Section 7.

2 Preliminaries

$\mathbb{N}$ denotes the set of natural numbers. A bag (or multiset) over some set S is a mapping $\mathcal{B} : S \to \mathbb{N}$ that maps each element to the number of times it occurs. The size of a bag $|\mathcal{B}|$ is the total number of elements in the bag including duplicates. $[a^2, b^4, c]$ represents the bag with $\mathcal{B}(a) = 2, \mathcal{B}(b) = 4, \mathcal{B}(c) = 1$ and

$\mathcal{B}(d) = 0$, where $a, b, c, d \in S$ and $|[a^2, b^4, c]| = 7$. The bag where all elements occur exactly once corresponds to a set. A sequence of length n over elements of set S is a mapping $\sigma \in \{1, \ldots, n\} \rightarrow S$. We denote the empty sequence by ϵ and non-empty sequences by listing their elements, e.g. $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$, where $e_i = \sigma(i)$ for $1 \leq i \leq n$. The size of a sequence $|\sigma|$ corresponds to the length of the sequence. $\sigma \uparrow S$ is the projection of σ onto elements of set S, e.g. $\langle a, b, e, a, b, c \rangle \uparrow \{a, c\} = \langle a, a, c \rangle$. The set of all sequences over S is denoted as S^*, the set of all sets over S as 2^S and the set of all bags over S as $\mathbb{N}^S$. The Parikh vector $parikh(\sigma) \in \mathcal{B}(S)$ denotes the number of occurrences of element s in a sequence σ, i.e. for $s \in S : parikh(\sigma)(s) = |\sigma \uparrow \{s\}|$. The $set : S^* \rightarrow 2^S$ is a function that transforms a sequence over S to a set of S, i.e. $set(\sigma) = \{a | a \in \sigma\}$. Functions can be composed by function composition. Let $f : A \rightarrow B$ and $g : B \rightarrow C$, then $g \circ f : A \rightarrow C$, such that $\forall x \in A : (g \circ f)(x) = g(f(x))$.

Current information systems log their activities (process steps) occurring in the context of the business processes they support. We assume that events in the log are uniquely associated with the activities in the process.

Definition 1 (Event). *Let $\mathcal{E}$ be the universe of events, i.e. the set of all possible event identifiers. Events are executed in the context of an instance of a process. Let $\mathcal{I}$ be the universe of process instance identifiers. We assume there is a function $pid : \mathcal{E} \rightarrow \mathcal{I}$ that maps each event to its process instance. Events can have additional parameters such as activity name, time stamp, executing resource and data attributes. We use $\mathcal{V}_\theta$ to denote the universe of values for a parameter θ. For each parameter θ we assume there exists a function π that maps an event to its parameter value, i.e. $\pi_\theta : \mathcal{E} \rightarrow \mathcal{V}_\theta$, e.g. $\pi_{cost} : \mathcal{E} \rightarrow \mathbb{Z}$.*

Events are linked (by *pid*) to a particular instance (or case) of a process. A log is basically a sequence over events from which event traces can be derived: A trace is an ordered sequence of events belonging to the same process instance where time is non-decreasing.

Definition 2 (Event trace). *An event trace is sequence of events $\sigma \in \mathcal{E}^*$, such that each event belongs to the same process instance, appears only once in the sequence and time is non-decreasing, i.e., σ is such that for $1 \leq i < j \leq |\sigma| : pid(\sigma(i)) = pid(\sigma(j))$, $\sigma(i) \neq \sigma(j)$ and $\pi_{time}(\sigma(i)) \leq \pi_{time}(\sigma(j))$. The universe of all event traces over $\mathcal{E}$ is denoted as $\mathcal{T}^{\mathcal{E}}$.*

In absence of time stamps, we assume events are ordered by their occurrence in the log, i.e. $\pi_{time}(\sigma(i)) = i$.

Definition 3 (Event log). *An event log (in the remainder referred to as log) is a set over event traces, formally $L \subset \mathcal{T}^{\mathcal{E}}$, such that each event occurs in at most one trace. $\forall \sigma_1, \sigma_2 \in L : set(\sigma_1) \cap set(\sigma_2) = \emptyset$ or $\sigma_1 = \sigma_2$.*

3 Adaptive Simulation Model

In this section we will elaborate on the definition of an adaptive simulation model that supports the (re-)estimation (adaptation) of simulation parameters during execution.

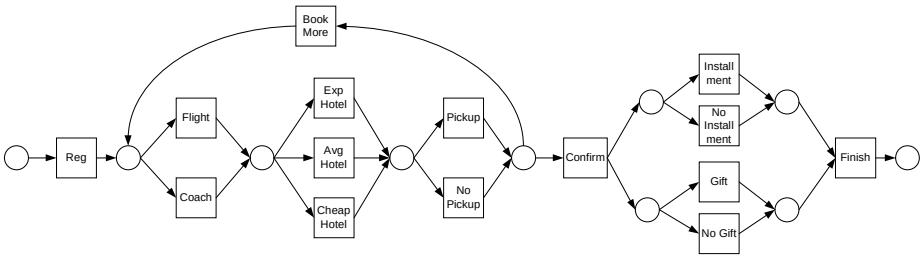


Fig. 1. The travel agency process.

As running example, we consider a simple travel agency process where customers can compose a trip by booking a flight or coach transport, and a hotel (luxury, middle class or budget), for one or more days. The trip can be composed of multiple transport-hotel combinations. For each hotel stay customers can make use of a pickup service that transports them to the city centre or the airport. Clients may choose to pay their holidays by installment. In addition customers who spend much money are rewarded with a gift. A good estimation for the number of pickups is necessary for arranging a suitable contract with one of the local taxi companies. The agency would also like to estimate the number gifts to be purchased. Early market research indicates a trend towards a decreasing budget for clients. The agency has disposal of a log containing information about customers of the past years. How will the expected trend affect the number of pickups and gifts for the next year?

Figure 1 depicts the control flow for the travel agency. It contains all typical routing constructs such as sequentiality, parallelism, iteration, synchronisation, deterministic and non-deterministic choices of the environment. For meaningful analysis parameters (e.g. activity durations and probabilities) should be added to the control flow that reflect the real execution of the process. Moreover, for accurate simulation, we need to incorporate existing execution dependencies into the model. We use the log of the process that contains past executions to detect dependencies and to estimate the simulation parameters.

Static models cannot capture correlations between the parameters, such as a decreasing probability to choose going on with booking after every iteration, or a correlation between the choice of transportation and the hotel class. This results in inaccuracies of the analysis, e.g. the estimation of the number of gifts that should be ordered.

In adaptive simulation models, we incorporate dependencies into the model. This allows simulation parameters to be updated during the simulation execution by considering the predictors that influence the parameter value and some equation describing how the simulation parameter changes in terms of the values of the predictors, e.g. the probability for booking an expensive hotel for customers who booked a flight is 60%, and for those who chose a coach it is 10%. Historical execution data, captured in a log, is used to find an equation that predicts the parameter value based on the value the predictor chosen for this

parameter. During the simulation, predictor values are derived from the trace of the running simulation instance, e.g. flight is the transportation type booked for some instance. The equations to determine the values of simulation parameters and the equations for deriving predictor values from the (prefix) of the simulation trace are included in the model. Note that each parameter can depend on multiple predictors and different parameters can have different predictors.

Note that predictors are not simply case variables that are defined by the designer of the process, nor are correlations the decision rules for the process. For example, the duration of an activity is the time that is actually needed for its execution rather than a predefined value. Different resources might need different time periods to execute the activity, which is also not something typically predefined.

In the remainder of this section we represent steps of the business process as activities in the simulation model, e.g. a (coloured) Petri net or any other formalism with clear execution semantics that allow for simulation. Activities can be associated to parameters such as cost, duration and execution probability.

Definition 4 (Model Parameters). *Let $\mathcal{A}$ be the universe of activities. The set of activities in model M is denoted as A_M, where $A_M \subseteq \mathcal{A}$. Activities can have additional parameters and we use Θ_M to denote the parameters of model M. The domain of parameter values of M is denoted as $\mathcal{V}_\Theta$. The values for the parameters of the model are stochastical values that we estimate based on a log (L) of some process associated to M.*

One can annotate the model with *fixed* values for each simulation parameter, following e.g. the approach proposed in [17].

Definition 5 (Annotated Static Model). *Let M be a model describing the relation between the set of activities $A \subseteq \mathcal{A}$. The static annotated version of M is described by $\mathcal{M}_s = (M, val_s)$ where $val_s : \Theta_M \rightarrow \mathcal{V}_\Theta$ is a function that maps parameters to parameter values.*

Parameters are mapped to the average value of this parameter in the log. The values of the static model are fixed, regardless the current simulation instance.

In adaptive models we assume that the values of simulation parameters depend on the state of the instance and can change during the development of the instance. A regression equation describes the relation between a response variable (here simulation parameter) and explanatory (or predictor) variables in a data set.

Definition 6 (Regression Equation for Parameters). *The regression equation for a parameter θ is a function $f_\theta : \mathbf{X} \rightarrow \mathcal{V}_\theta$, describing the response of parameter θ to some experimental setting specified by a vector of predictor variables $\mathbf{x} \in \mathbf{X}$. We use $\mathcal{R}$ to denote the universe of regression equations.*

The selection of f_θ is a choice that should be carefully matched with the data set and assumptions on the data. A general additive multiple regression model

which relates a dependent variable y to k predictor variables $x_1, x_2, \ldots, x_k$, is given by the model equation $y = a + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + e$, where a is the intercept, e is random deviation and each β_i is a population regression coefficient for predictor x_i [7]. In this model the right hand side of the equation is the population regression function (f_θ) that determines the outcome given a vector of predictor variables $\mathbf{x}$. Qualitative predictors variables (e.g. the name of an activity) can be encoded [7]. To predict the probability we use the generalized logit model for multinomial response [3]. Statistical packages such as R [13], a software environment for statistical computing and graphics, contain functionality for encoding data and fitting models on data.

Definition 7 (Predictor Value Function). *The predictor value function ϑ is a function that maps a trace to a vector of predictor values, $\vartheta : \mathcal{T}^{\mathcal{E}} \rightarrow \mathbf{X}$. We use $\mathcal{V}$ to denote the universe of predictor value functions.*

The predictor value function extracts the value of the predictor from a partial trace. For example for pickup probability in Figure 1 the predictor value function could be defined as $\vartheta(\eta) = [lastHotel(\eta), lastTravel(\eta)]$, where η is the partial trace of the instance, *lastHotel* determines the type of the last hotel that was booked and *lastTravel* determines the type of the last travel that was taken.

Definition 8 (Annotated Adaptive Model). *Let M be a model describing the control flow based on the selection of activities $A \subseteq \mathcal{A}$. The adaptive annotated version of M is described by $\mathcal{M}_a = (M, val_a, regr, pred)$ where $val_a : \Theta \times \mathcal{T}^{\mathcal{E}} \rightarrow \mathcal{V}_\Theta$ is a function that maps parameters to parameter values, depending on a (partial) simulation trace, $regr : \Theta \rightarrow \mathcal{R}$ is a function that maps parameters to corresponding regression functions and $pred : \Theta \rightarrow \mathcal{V}$ maps a parameter to a predictor value function. Assume $regr(\theta) = f_\theta$ and $pred(\theta) = \vartheta$. Then the parameter value for θ, given trace $\eta \in \mathcal{T}^{\mathcal{E}}$, is defined as $val_a(\theta, \eta) = f_\theta \circ \vartheta(\eta)$.*

Values for parameters of the adaptive model can be obtained by applying the associated regression equation on the current predictor values.

Example 1 (Adaptive Parameter Value). Consider again the travel agency process depicted in Figure 1. From the log we derive a regression equation for the pickup probability parameter. Suppose the regression equation to estimate the pickup probability is based on the type of the last booked hotel and the type of the last travel. The predictor value function extracts this property from the instance history. The parameter is estimated by applying the regression function on the current predictor values obtained from the current simulation trace, i.e. $val_a(prob\_Pickup, \eta) = f_{prob\_Pickup} \circ \vartheta(\eta)$, where the predictor value function is defined by $\vartheta(\eta) = [lastHotel(\eta), lastTravel(\eta)]$ and extracts the type of the last booked hotel and the type of the last travel from the trace. The regression function f_{prob_Pickup} predicts the value of the pickup probability, given the last type of hotel and travel.

4 Mining Dependencies

For model M with parameters Θ_M and log L we mine an adaptive simulation model $\mathcal{M}_a = (M, val_a, regr, pred)$, where we set the adaptive parameter value for each parameter according to the current simulation trace η using the predictor value function and the regression function. Using regression analysis we can find dependencies between parameter and predictor values.

What can be suitable predictor candidates in terms of event traces of business processes? An obvious predictor candidate seems to be the partial trace of the instance. For real-life processes however, the log contains a wide variety of traces, but typically not many of them follow the same execution scheme and not all possible traces are contained in the log. As we showed in [18], trace abstractions can be applied to tackle this issue. Such abstractions consider some characteristics of the trace rather than the exact trace; the occurrence of a single activity, or a choice that was made at some point in the process are examples of abstractions that can be used as predictor. The goal is to find those trace characteristics (captured by a trace abstraction) that are good predictors for a simulation parameter. We do this by applying existing statistical methods where we define regression models for different predictor combination and determine which regression model fits the data set best. During simulation, parameters in the adaptive simulation model are determined by the associated regression model, based on the abstraction values for the current simulation trace.

Abstractions we consider are functions that map one representation of a partial trace to another, leaving out irrelevant details. Regression analysis is used to detect which abstractions are good predictors for a parameter. The input values for the regression equation are defined by the predictor value function ϑ (cf. Definition 7).

Definition 9 (Predictor values for Abstractions). *Let the predictors for a regression equation be given by a vector of k abstractions $[\alpha_1, \ldots, \alpha_k]$. Then for all traces $\eta \in \mathcal{T}^{\mathcal{E}}$ the input for the regression equation is defined as $\vartheta(\eta) = [\alpha_1(\eta), \ldots, \alpha_k(\eta)]$, where $\alpha_i(\eta)$ denotes the value of the i^{th} abstraction applied to η.*

4.1 Abstractions

This section presents an overview of elementary abstractions and compositions thereof.

Property projection ($property_p^{\alpha} : \mathcal{T}^{\mathcal{E}} \rightarrow \mathcal{V}^*$) converts a sequence of events to a sequence of their properties, e.g. a sequence of data attributes or time stamps, i.e. $property_p^{\alpha}(\langle a_1, \ldots, a_n \rangle) = \langle \pi_p(a_1), \ldots, \pi_p(a_n) \rangle$.

Event projection is an abstraction that can be used to extract specified elements from a trace. Event projection is a function ($event_A^{\alpha} : \mathcal{V}^* \rightarrow \mathcal{V}^*$) that retains elements of $\sigma \in \mathcal{V}^*$ that are in A, i.e. $event_A^{\alpha}(\sigma) = \sigma \uparrow A$.

Window abstraction defines the region of interest within a trace as subtrace. Window abstraction takes some (or all) consecutive elements of the trace ($window^{\alpha} : \mathcal{V}^* \rightarrow \mathcal{V}^*$). The window can be specified by an interval between two points (denoted as $window^{\alpha}_{P_1,P_2}$), or by one point, a direction and a width (denoted as $window^{\alpha}_{P,d,w}$). A point P can be a concrete event (e.g. the last occurrence of an event with a certain event name) or it can be characterised by some condition on the event (e.g. the i^{th} event from the trace). The direction d of the window is specified prior to ($<$) or after ($>$) the point. The width w of the window is specified by a time interval or some condition.

Bag, set and cardinality abstractions Abstraction from the event ordering can be done by the bag abstraction. Set abstraction abstracts both from event ordering and their frequencies. Cardinality abstraction can be used to focus on the size of sequences, sets and bags.

Bag abstraction ($bag^{\alpha} : \mathcal{V}^* \rightarrow \mathbb{N}^{\mathcal{V}}$) is a function that transforms a sequence $\sigma \in \mathcal{V}^*$ into a bag, i.e. $bag^{\alpha}(\sigma) = par(\sigma)$.

Set abstraction ($set^{\alpha} : \mathcal{V}^* \rightarrow 2^{\mathcal{V}}$) is a function that transforms a sequence $\sigma \in \mathcal{V}^*$ into a set, i.e. $set^{\alpha}(\sigma) = set(\sigma)$.

Cardinality abstraction ($cardinality^{\alpha} : \mathcal{C} \rightarrow \mathbb{N}$) is a function that gives the size of a collection $\mathcal{C}$, where $\mathcal{C}$ is $\mathcal{V}^*$, $2^{\mathcal{V}}$, or $\mathbb{N}^{\mathcal{V}}$, i.e. $\forall c \in \mathcal{C} : cardinality^{\alpha}(c) = |c|$.

Last occurrence abstraction considers the last occurring element from a specified set. This abstraction allows us to look, for example, at the most recent value of a data element associated with an activity that can re-occur in an iterative process, such as the last test outcome in Figure **??**.

Last occurrence abstraction ($last^{\alpha}_{A} : \mathcal{V}^* \rightarrow A \cup \{\perp\}$) is a function that gives the last occurring element of set A in trace σ over E, if any, otherwise undefined $\perp$.

$$last^{\alpha}_{A}(\sigma) = \begin{cases} t \text{ , if } \exists t \in (A \cap E), \gamma \in E^*, \delta \in (E \setminus A)^*, \text{ such that } \sigma = (\gamma; t; \delta) \\ \perp \text{ , if } \sigma \in (E \setminus A)^*. \end{cases}$$

Existence abstraction detects the occurrence of a specified event. For example, for an insurance company the probability that the extensive procedure for handling claims will be chosen is higher if the person has already committed fraud. Existence abstraction is a function ($existance^{\alpha}_{e} : \mathcal{V}^* \rightarrow \{true, false\}$) that indicates whether element e is part of a sequence, i.e. $existance^{\alpha}_{e}(\sigma) = e \in \sigma$.

Duration abstraction can be used to obtain the duration between two events, e.g. the shorter the test procedure is, the more likely its result will be negative and another repair try will be needed. Duration abstraction ($duration^{\alpha}_{e_1,e_2} : \mathcal{V}^* \rightarrow \mathbb{N}$) is a function that indicates the time duration between two events e_1 and e_2, i.e. $duration^{\alpha}_{e_1,e_2}(\sigma) = \pi_{time}(event^{\alpha}_{e_2}) - \pi_{time}(event^{\alpha}_{e_1})$.

In practical applications combinations of abstractions, constructed as function compositions, are often used.

Example 2 (Combination of abstractions). The number of iterations alreasy taken when executing a process from Figure 1 can be computed from a partial trace

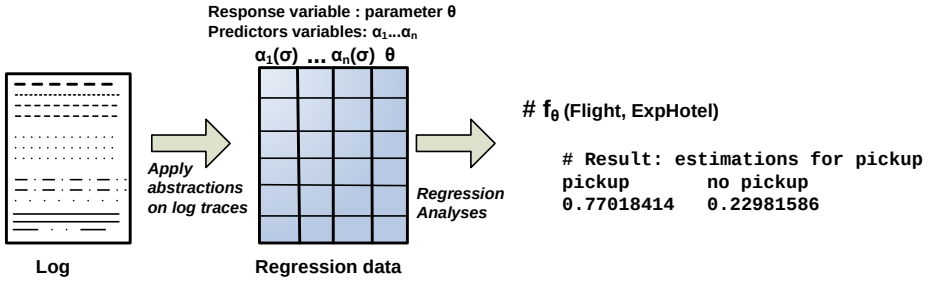


Fig. 2. Mining predictors (abstractions).

as the number of occurrences of the *BookMore* transition, i.e. $cardinality^{\alpha} \circ event^{\alpha}_{\{BookMore\}}(\sigma)$.

4.2 Mining Predictors

Section 4.1 identifies a collection of possible abstraction candidates that can be used to predict a parameter. For the selection of suitable predictors for a parameter we can define regression models with different combinations of predictors and find the model that best fits the data set that is derived from the execution log. In fact the execution log is converted into a list of parameter observations where each observation contains the value of the parameter and the values of all predictors under consideration. The predictor values for a parameter can be obtained by applying each predictor on the prefix of the parameter. The data set consists of the observations of all log traces for a parameter and a collection of abstractions.

Suppose, we want to observe the probability of selecting option c_1, c_2 or c_3 (in a free choice construct) where we consider the set abstraction set^{α} and cardinality abstraction $cardinality^{\alpha}$, which is used to count the number of times activity a occurs. Consider the observations for log trace $\sigma = \langle a, b, e, c_1, b, f, c_3, a, b, f, c_1 \rangle$. In this trace there are three observations, one for each occurrence of c_1, c_2 or c_3. For the first occurrence of c_1 the observation is $[\{a, b, e\}, 1, c_1]$, determined by the set^{α} and $cardinality^{\alpha}$ on prefix $\langle a, b, e \rangle$, similarly we can observe $[\{a, b, e, f\}, 1, c_3]$ and $[\{a, b, e, f\}, 2, c_1]$.

The data set, which can be obtained by traversing the log once, is the input for regression analysis. There are different methods to systematically determine the model with the best combination of predictors for a data set. One can stepwise eliminate or add predictors to the model based on statistical relevance with respect to a reference model, or define all models, based on the power set of predictors, and find the best fitting model. In the adaptive simulation model the best fitting regression model is used for predicting the parameter value. Recall that the predictors of a regression model are in fact abstractions. Applying these abstractions on the simulation trace yields the predictor values for the regression model.

5 Experiments

In this section we illustrate and validate our approach by using adaptive simulation models to analyse the effect of trends on a business process. Since it is infeasible to expose a real-life process to trends for the purpose of validating our approach, we conduct our experiments on a reference model which is based on the control flow given in Figure 1. From the log of the reference model (M_r) we derive a static simulation model (M_s) and an adaptive simulation model (M_a). The three models are then exposed to the same trend and the results are compared.

5.1 Experiment Set-up

For the experiments we define a reference model M_r that emulates a real process and produces logs. From these logs we build a static M_s and an adaptive simulation model M_a. To evaluate the suitability of the simulation models for evaluating business trends we expose the business process (the reference model) and the simulation models to some trends, i.e. a change in the environment of the processes such as a customer bias towards cheaper transportation. We compare the capabilities of the adaptive and static model by comparing their performance with respect to the performance of the reference model. The performance is measured in occurrence ratio of *Pickup* and *Gift* transitions.

Reference Model. The role of the reference model is to produce logs by emulating a real life process and its environment. A complex stochastic scheme has been added to the control flow depicted in Figure 1 in order to equip the process with non-trivial dependencies, so that it became impossible to analytically compute dependencies between different choices and parameter values. In real life these dependencies are not known; in the reference model we in fact use them to emulated human decisions. Furthermore the reference model has been equipped with logging functions that log every activity that is executed during simulation.

We run 30 replications of 10.000 instances on M_r, each replication creating a log. We randomly select one of these logs to detect dependencies and to create an adaptive model M_a and a static model M_s. To create these models one can annotate the *control flow* of the travel agency process based on the log of M_r. (Note that the control flow can also be mined from this log by standard process mining techniques.) It is important to note that these models are created based on the log without using any knowledge of the stochastic scheme of M_r.

Static Model We mine the log of M_r to find the values for the parameters of the static model M_s, using techniques from [17]. From the log of M_r we can determine such parameters as the probability to book an expensive hotel as the percentage of cases in which expensive hotels were booked and annotate the control flow to obtain the static model, cf. Def. 5. Recall that in the static model the values are fixed and that the partial simulation trace (or the history) is not used to estimate their values.

Adaptive Model For the adaptive model we use the log of M_r to determine the regression model that best fits the log, cf. Section 4. We annotate the control flow model with the obtained model, and the predictors (abstractions), cf. Def. 8, so that the parameter values can now be determined during the simulation, depending on the partial simulation trace.

Mining Simulation Model Parameters. The probabilities for the static model for firing the *Pickup* and the *Gift* transitions can be mined using the `Performance Analysis with Petri net` plugin of ProM [1]. For the adaptive model parameters we determine suitable predictors for those transitions with regression analysis using the `multinom` function from the `nnet` library in R [13]. The R data is created using our ProM `R Data` plugin [1]. This function fits multinomial logit models with nominal response categories.

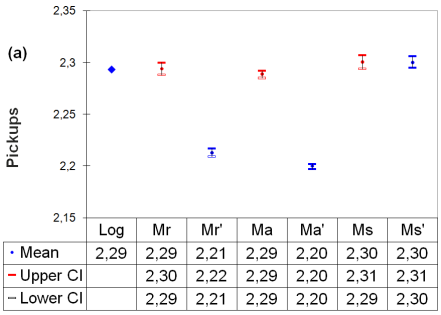
Injecting the Trend. To evaluate the suitability of the simulation models for evaluating business trends we expose the reference model to a lower customer budget, affecting the way of travelling, hotel type and number of composed travel-hotel combinations. On the reference model the trend results in decreasing the share of flights from 50% to 10%. The shares of expensive, medium and cheap hotels change from 33% to 10%, 25% and 65%, respectively. Finally, the number of booked combinations drops. The trend of booking less flights is injected in the simulation models in a consistent manner; the bounds for the guards that control the transportation choice are set to the mentioned probabilities. Note that, except for the injected trend, the simulation models do not change.

Running the simulations. All models have been implemented as coloured Petri net in CPN tools [11], which is a well established tool for modelling and analysis of Coloured Petri Nets. We refer the interested reader for implementation details of the models to [12]. For each model we run 30 replications of 10.000 instances. For each replication we determine the occurrence ratio of *Pickup* and *Gift* transitions. The result of all the replications are depicted as confidence intervals.

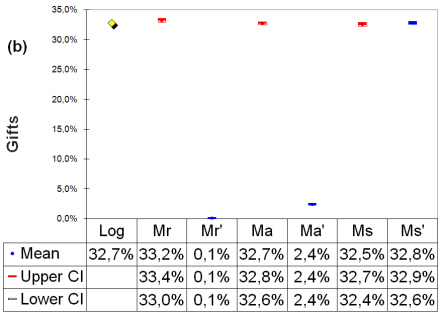
5.2 Results

The procedure to mine correlations starts with the definition of the response variable and the predictors for the estimation for the response variable.

Estimating the Pickup Probability. To convert the log into regression data, the response variable and the predictor variables have to be chosen. For the probability of *Pickup* we consider the following abstractions: (1) which hotel was the last one that was booked, (2) last type of travel and (3) the number of iterations. We convert the log for these abstractions to a data format that can be used for regression analysis. The data is imported into R where we fit the data using `multinom` function from the `nnet` library. For each combination of abstractions we defined a logit model. From the models with a single predictor, the model



	Log	Mr	Mr'	Ma	Ma'	Ms	Ms'
Mean	2,29	2,29	2,21	2,29	2,20	2,30	2,30
Upper CI		2,30	2,22	2,29	2,20	2,31	2,31
Lower CI		2,29	2,21	2,29	2,20	2,29	2,30



	Log	Mr	Mr'	Ma	Ma'	Ms	Ms'
Mean	32,7%	33,2%	0,1%	32,7%	2,4%	32,5%	32,8%
Upper CI		33,4%	0,1%	32,8%	2,4%	32,7%	32,9%
Lower CI		33,0%	0,1%	32,6%	2,4%	32,4%	32,6%

Fig. 3. Simulating the effect of clients becoming more poor. M_r is the reference model, M_a is the history-dependent model and M_s a history-independent model. After inserting the new trend referred to as M_r', M_a' and M_s'. (a) Depicts the effect on the number of *Pickup* using the last occurrence of the hotel type as predictor and (b) depicts the effect on percentage clients that get a *Gift*, using the number of iterations as predictor.

that considered the last hotel fitted the data best. Moreover, adding more predictors did not significantly improve the results. Therefore, only the last hotel was used for estimating the response probabilities for all predictor combinations, e.g. the probability for *Pickup* given that the last hotel was an expensive hotel is 53%, for a budget hotel this probability is 44%.

The conversion of the log took 1 minute and 24 seconds and the execution of all R commands (including the fitting and testing of other models) took 39 seconds.

Estimating the Gift Probability. Similarly we determine the predictors for estimating the probability for gifts. Different abstractions and combinations thereof have been considered, including (1) the set abstraction on the hotels, (2) the sequence of the last two hotels, (3) the set abstraction on the travel types and (4) the sequence of the last two travel types and (5) the number of booked combinations. The model using the last abstraction as predictor was the one with the best fit.

Simulated Process Performance. Figure 3 depicts the results of simulation of the different models before and after the injected trend of customers with a lower budget. The results are depicted as 95% confidence intervals, that depict the occurrence ratios of the *Pickup* and the *Gift* transitions. On the left side of the figures we depict the value for the log produced by M_r that was (randomly) selected to derive the adaptive M_a and static model M_s. The confidence intervals shown for M_r, M_a and M_s are based on simulation without introducing the trend; here both M_a and M_s approximate the behaviour of M_r well. After the trend is introduced, only M_a is able to follow the direction of the trend whereas M_s is unable to do so, because essential correlations are not taken into account as all choices are considered mutually independent.

Notice that M_a slightly underestimates the number of pickups. This is caused by the fact that the log randomly selected from 30 replications contains slightly less pickups than obtained on all the 30 logs on average. Also note that M_a overestimates the number of gifts for the new situation due to the fact that we try to capture a complex data dependency by a very simple abstraction, which does not exactly captures the dependency but approximate it.

It is clear the the adaptive model gives a much better approximation than the static model. Our experiments show that the history-dependent mechanism adapting simulation parameters according to the developments in the running instance is able to propagate environmental changes in the simulation. For logs containing data, abstractions on data can be used to obtain even more precise results.

6 Related Work

Business Process Simulation (BPS) has been indicated by [10] as an essential technique for Business Process Re-engineering (BPR) where it is not only important to understand the static behaviour of the process, but also to accurately predict the outcome of proposed and/or expected changes for the process to judge the effect on the organisation performance. This does not only apply to the area BPR, which traditionally focuses on complete redesign of existing processes, but it is also interesting in a less radical setting: *"How will a trend affect the performance of my existing business process?"*. Simulation offers support for randomness, uncertainty and interdependencies, making it a valuable technique for business process management.

The biggest challenge in the development of a simulation model is obtaining an accurate model that is close to reality. To tackle the problem of creating realistic simulation models, [17] present a method to generate simulation models based on actual information from logs. The authors create simulation models in CPN tools [11] capturing the control flow perspective, the resource perspective and the data perspective and the current state. In their approach they assume all variables to be independent. This assumption is, however, unrealistic for real-life business processes, as [5] explains, dependencies and correlations present in business processes.

History-dependent Petri nets [18] (HDSPNs) are an extension of classical Petri nets [6, 16] that use a history-dependent mechanism to model history-dependent choices. This approach can easily be extended to model cost and dependencies for activities.

Detection of correlations from process logs and using them for business process simulation has not been studied extensively yet. Usually assumptions are made about the dependency and/or distributions of the variables [4,15,17]. Correlations between quantitative variables in business processes can be used to obtain more accurate settings for cost and durations of variables. These correlations can be derived from data with simple statistical techniques. When qualitative variables (e.g. resource and activity names) are involved, more advanced tech-

niques such as regression analysis are required. Closest related work is the one on predictions in business processes [8]. There non-parametric regression analysis is used to predict the finishing time of an instance. Intermediate process dependencies are not considered. The complexity of constructing the regression model is a serious limitation of that approach.

7 Conclusion

In this paper we focused on analysing the effects of trends on existing business processes. The analysis is performed on simulation models based on information obtained by actual executions (a log) of the process. The main idea is that dependencies need to be included into the simulation model to accurately predict the global effect of new trends. For this purpose we introduce adaptive simulation models that have simulation parameters (re-)estimated during execution. We discussed how dependencies can be derived from a log of a business process using regression analysis. We use abstractions on traces to balance between the amount of data available in the log and the amount of information necessary to make good predictions. The conversion of log data to R data, for selected abstractions candidates is implemented as the `R Data` plugin in the ProM framework [1]. This data can used directly in R to determine the best fitting regression model.

The obtained dependencies are included into the simulation model by means of a history-dependent mechanism that uses the partial simulation trace to determine further simulation parameters. We have demonstrated the application of dependencies in adaptive simulation models on a reference model from whose log we created an adaptive and a static simulation model. Only the adaptive simulation model was able to propagate the trend into the correction direction.

An important direction for future work goes into the direction of the generation of R data given some abstractions. Currently abstractions can have many values (levels) and the generated data can be sparse, making it unsuitable for regression. As future work we plan to look into data mining techniques to cluster abstraction levels. Furthermore, we will focus on doing experiments that consider the probability, cost and duration of parameters and predictors and where more complex abstraction compositions are considered.

References

1. *ProM Nightly Builds*, 2006. http://prom.win.tue.nl/tools/prom/nightly/.
2. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
3. A. Agresti. *Categorical Data Analysis*. Wiley Series in Probability and Statistics. Wiley-Interscience, 2nd edition, 2002.
4. F. Baccelli and P. Konstantopoulos. Estimates of Cycle Times in Stochastic Petri Nets. *Rapport de recherche 1572, INRIA, Rocquencourt*, 1992.

5. A.P. Barros, G. Decker, and A. Grosskopf. Complex Events in Business Processes. In *BIS*, volume 4439 of *Lecture Notes in Computer Science*, pages 29–40. Springer, 2007.
6. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.
7. J. Devore and N. Farnum. *Applied Statistics for Engineers and Scientists*. Duxbury, 1st edition, 1999.
8. B.F. van Dongen, R.A. Crooy, and W.M.P. van der Aalst. Cycle Time Prediction: When Will This Case Finally Be Finished? In *CoopIS 2008, OTM 2008, Part I*, volume 5331 of *Lecture Notes in Computer Science*, pages 319–336. Springer-Verlag, 2008.
9. S. Ferson and M.A. Burgman. Correlations, Dependency Bounds and Extinction Risks. *Biological Conservation*, 73(2):101 – 105, 1995. Applications of Population Viability Analysis to Biodiversity.
10. B. Gladwin and K. Tumay. Modeling Business Processes with Simulation Tools. In *WSC '94: Proceedings of the 26th conference on Winter simulation*, pages 114–121, San Diego, CA, USA, 1994. Society for Computer Simulation International.
11. K. Jensen, L.M. Kristensen, and L. Wells. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. *International Journal on Software Tools for Technology Transfer*, 9(3-4):213–254, 2007.
12. J. Jian. Mining Simulation Models with Correlations. Master's thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2009.
13. R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006. ISBN 3-900051-07-0.
14. G. Regev and A. Wegmann. Why Do We Need Business Process Support? Balancing Specialization and Generalization with BPS Systems (Introductory note). In *CAiSE Workshops*, 2003.
15. H. A. Reijers. Case Prediction in BPM Systems: A Research Challenge. *Journal of the Korean Institute of Industrial Engineers*, 33:1–10, 2006.
16. Wolfgang Reisig and Grzegorz Rozenberg, editors. *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, volume 1491 of *Lecture Notes in Computer Science*. Springer, 1998.
17. A. Rozinat, R. S. Mans, M. Song, and W. M. P. van der Aalst. Discovering Simulation Models. *Inf. Syst.*, 34(3):305–327, 2009.
18. M.H. Schonenberg, N. Sidorova, W.M.P. van der Aalst, and K.M. van Hee. History-Dependent Stochastic Petri Nets. In *Perspectives of System Informatics (PSI 2009)*, volume 5947 of *LNCS*, pages 366–379. Springer, 2009.