Enabling Probabilistic Process Monitoring in Non-automated Environments

Andreas Rogge-Solti and Mathias Weske

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

Abstract. Business processes are crucial for every organisation as they represent the core value generating processes. Managing business processes is important to be efficient and to compete with a globalized market. Business process monitoring is an essential means to understand and to improve working procedures. It helps detecting deviations from planned procedures and brings transparency into the state and progress of running process instances. However, without automated execution of business processes via a workflow engine, the absence of execution information hampers monitoring. Often, the automated execution of business processes is neither feasible nor desirable. However, a few monitoring points can be used, when process participants interact with IT-systems.

In this paper, we propose a novel approach to business process monitoring using probabilistic estimations to fill information for missing monitoring points. The applicability of the approach is evaluated with a case study in a German university hospital.

Keywords: business process monitoring, probabilistic analysis, sparse events.

1 Introduction

Monitoring of business processes and process activities [1] emerged as an instrument in business process management to gain insights into running process instances. It allows for live evaluation and keeping track, whether the process executes as planned. It can also be used to predict resource shortages and provide decision support for timely operational changes in process instances [2].

However, current business process monitoring solutions require detailed log files containing entries for each monitored activity in the process, often realized by workflow enactment components that emit events for each activity.

Another method that can be used for correlating live event streams to process models is process mining and it does not depend on complete log files and can deal with noise, i.e., missing or multiple events. However, it is used mainly for conformance checking between process models and logs, by computing *fitness* and *appropriateness* [3]. The difference to our approach is that it does not deal with untracked process fragments, i.e., parts of processes that *never* occur in the logs.

We want to address this gap and provide a monitoring approach for such use cases. More concretely, we enable monitoring of untracked parts. The approach is based on

I. Bider et al. (Eds.): BPMDS 2012 and EMMSAD 2012, LNBIP 113, pp. 226–240, 2012.

probabilities; it allows to compute the probable state of the unmonitored parts of the process for a given time. We resort to an analytical model, i.e., calculating the expected behaviour of the process parts, where no events are available that indicate process progress. The question we want to answer in this paper is: *Which activities in the process are probably being executed at a given time?*

If we know the answer to this question, we can also derive an answer to the question: What is the probability of finishing the process in a given time duration? It also might help to know that there is a (high) chance that a process instance will be performing a task in near future that requires a specific resource.

The applicability of the approach is evaluated in a real world case study in a university hospital in Germany. In hospital environments, or more generally, if people perform manual activities in processes, the utilization of workflow enactment systems is neither feasible nor desirable [4]. However, some important events might be available in IT-systems nonetheless. This is the case in the liver transplant surgery procedure in the Jena University Hospital, where several important events during the surgery, e.g., performing the cut, discharge from the room, etc. are recorded in the hospital information system.

The remaining paper is organized as follows. Sect. 2 introduces preliminary definitions and notions required for describing the scenario and approach formally. Next, in Sect. 3, we describe the related work with focus on analytical models, process mining and monitoring systems. Sect. 4 describes the proposed probabilistic approach to monitoring. In Sect. 5, the applicability of the approach is evaluated with a case study in the Jena University Hospital. Sect. 6 concludes the paper and discusses limitations and future work.

2 Preliminaries

In this section, we give a formal description of the setting in order to describe the monitoring approach explicitly. Therefore, we first explain the model level by introducing the notions of process model, activity lifecycle, and events. Then, we relate these notions to the corresponding instance level, i.e., the notion of process instances and events.

The elicited process models for the clinical pathways contain information about the activities that need to be performed by the clinical personnel. The process flow can be sequential, or split into mutually exclusive or parallel branches at gateways that can also be used to merge alternative or parallel flows. Thus, we use the following definition of a process model. (Let $\dot\cup$ be the disjoint union.)

Definition 1 (Process Model). *Let $\mathcal{P}$ be the set of process models. A process model $P \in \mathcal{P}$ is a connected, directed, acyclic graph (N, E) where $N = A \,\dot\cup\, G \,\dot\cup\, \{n_{in}, n_{out}\}$ is a finite set of nodes, with activities A, gateways $G = G_{XOR} \,\dot\cup\, G_{AND}$, with exactly one start event n_{in} and one end event n_{out}, and $E \subseteq (N \setminus \{n_{out}\}) \times (N \setminus \{n_{in}\})$ is a set of edges connecting the nodes.*

Each node $n \in N$ lies on a path from n_{in} to n_{out}. There are no loops allowed in the process model (acyclic property) and there is exactly one edge entering and leaving an activity, i.e., $\forall a_i \in A, \forall n_i, n_j \in N : ((n_i, a_i) \in E \wedge (n_j, a_i) \in E) \Rightarrow$

$n_i = n_j \wedge ((a_i, n_i) \in E \wedge (a_i, n_j) \in E) \Rightarrow n_i = n_j$. *Gateways can be* either *of branching or of merging nature.*

We assume that the process models are sound*, c.f., [5].*

The branching and merging nature of gateways is just a syntactical restriction, because every gateway that is both, can be split into two sequential gateways, one gateway merging the branches and a second one splitting them.

This set of modeling capabilities suffices for modeling most processes [6] and more complicated modeling constructs are not necessary to illustrate the proposed techniques.

Each activity $a_i \in A$ follows an activity lifecycle described by several states and state transitions. While detailed activity lifecycles were introduced in literature [7,8], we resort to a basic one for our monitoring purposes.

Definition 2 (Activity Lifecycle Model). *The activity lifecycle model is described by the states* $S = \{init, enabled, running, terminated\}$*, where* $init$ *is the start state, and the corresponding set of transitions* $T = \{begin, enable, terminate\}$ *change the states of the activity lifecycle according to the transition functions* $t : T \times S \rightarrow S$*. Where* $t(enable, init) = enabled$*,* $t(begin, enabled) = running$ *and* $t(terminate, running) = terminated$*.*

The state diagram in Fig. 1 depicts the possible states and transitions in the activity lifecycle model.

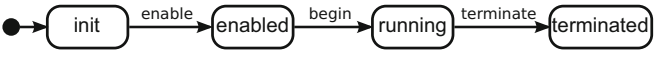


Fig. 1. State transition diagram of the activity lifecycle

Definition 3 (Possible Monitoring Point). *The set of possible monitoring points* PMP *for each process model* P *is defined as follows:* $\forall P \in \mathcal{P} : PMP(P) = A \times T$*, s.t.* A *is the set of activities in* P *and* T *is the set of transitions in the activity lifecycle. This means that every state transition of each activity in a process model is a possible monitoring point.*

We assume that the traversal of edges and gateways (except the joining gateway of parallel branches) does not take time. Therefore, the $terminated$ state of activities is linked to the $init$ state of the immediately following activities. Immediately following means either there is an edge between two activities a_1 and a_2, or there is a path from a_1 to a_2 on which there are no activities, but gateways. The semantics of this link is, that once the $terminated$ state of one activity is reached, the linked activities enter their $init$ state. Fig. 2 shows the combined activity lifecycle of two sequential activities a_1 and a_2. The link is depicted as dotted arrow between a_1 and a_2. Fig. 2 also shows the possible monitoring points at the state transitions of the activities lifecycle.

In order to specify the events triggering the state transitions in the activity lifecycle of a certain process instance, we first need to introduce the notions of event type and process instance.

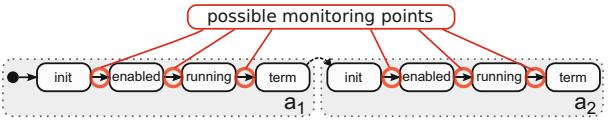


Fig. 2. Possible monitoring points in the process

Definition 4 (Event Type). *Let* $et_1 \dots et_n \in EType$ *be a finite set of event types.*

An example for this set is EType $= \{et_1$=*"new patient entry stored"*, et_2=*"surgery scheduled"*, et_3=*"anestetic given to patient"*, et_4=*"patient enters surgery room"*, et_5 = *"patient left surgery room"*$\}$ as in Fig. 3.

Definition 5 (Process Instance). *Let* $\mathcal{I}$ *be the universe of process instances. An instance* $i \in \mathcal{I}$ *represents a concrete case.*

We assume that events are characterized by various *properties*, e.g., an event has a time stamp and has a certain event type; it is executed by a particular person and is associated with a process instance. We do not restrict the set of properties, but assume, that the properties time stamp, event type, and process instance identifier exist for all events, c.f. [9].

Definition 6 (Event, Property). *Let* $\mathcal{E}$ *be the event universe, i.e., the set of all possible event identifiers and* $\mathcal{T}$ *the time domain. There are property functions* $prop_T : \mathcal{E} \rightarrow \mathcal{T}$ *assigning time stamps to events,* $prop_{EType} : \mathcal{E} \rightarrow EType$ *assigning event types to events and* $prop_I : \mathcal{E} \rightarrow \mathcal{I}$ *assigning instances to events.*

In the case that a workflow system [10] is executing a process model, i.e., the model is instantiated and traversed step by step, the generation of events marking all the state transitions of every visited activity can be done by the workflow system automatically. Thus, the alignment of these generated events to the model is straight-forward and monitoring, which activity is currently running becomes a trivial task. If, however, no workflow system is executing the model, the first challenge is to align corresponding process related events to the model. This is done through the definition of monitoring points in the model.

Definition 7 (Monitoring Point). *Let* P *be a process model,* $PMP(P)$ *represents possible monitoring points in* P*. Monitoring point* $m : PMP \nrightarrow EType$ *is a partial function mapping a possible monitoring point in the process model* P *to an event type.*

Note that m is a partial function, i.e., only defined for a subset of the input, because usually not every transition in the activity lifecycle of activities in a process is captured by events in underlying systems.

The idea is that the monitoring system subscribes to all events defined corresponding to the monitoring points. Let P be a surgery process model with activities $A = \{a_1$ = *"Schedule surgery"*, $\dots$, a_n = *"Perform stationary care"*$\}$ as in Fig. 3. The monitoring points function m in the example in Fig. 3 maps the $terminate$ transition of the activity lifecycle of activity *"Schedule surgery"* to event type *"surgery scheduled"* = et_2, i.e., $m((a_1, terminate)) = et_2$.

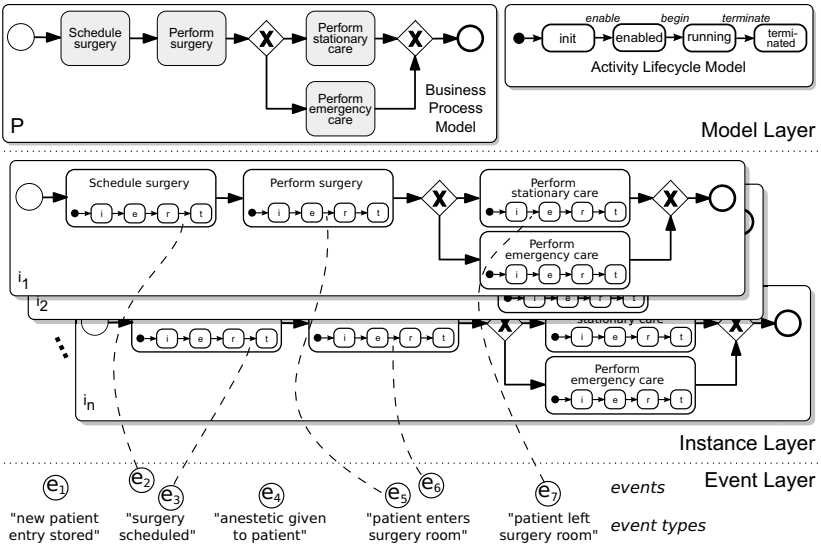


Fig. 3. Simple treatment process model with activity lifecycles for each activity. Dashed lines indicate the monitoring points available in the model, that are fed by events in the surrounding IT-environment.

In this case, the monitoring system subscribes to the event types et_2, et_4, and et_5. Whenever an event $e_i \in \mathcal{E}$ with one of these types occurs in the event layer, the corresponding process instance with identifier $i = \text{prop}_I(e_i)$ is updated by firing the state transition of the activity specified by m. This is illustrated in Fig. 3 by the dashed lines connecting activity state transitions of process instances with events. Note, that not every event is mapped to an activity state transition and neither do events for every activity state transition exist.

For probabilistic monitoring, we further need a model for the behaviour of the system over time, i.e., we need a way to express when activities will be completed. As we cannot know that exactly, we could specify a range over time based on our observations, or more generally specify a distribution of activity durations. In our model, we assume that activity durations, i.e., the duration from *enable* to *terminate*, can be characterized by the normal distribution, as in [11]. Even if, as indicated in [12], time durations are rather log-normally distributed (or follow another distribution type), the proposed concepts and computation mechanisms of the durations remain the same. However, calculating the sum of log-normally distributed activities can then only be approximated numerically [13].

Definition 8 (Activity Duration). *Each activity $a \in A$ in the process model P is annotated with an estimated normal probability distribution $\delta(a_i) = \mathcal{N}(\mu, \sigma^2)$, with the probability density function*

$$f_a(t) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(t-\mu)^2}{2\sigma^2}} \tag{1}$$

The cumulative distribution function $F_a(t) = \int_0^t f_a(t)\mathrm{d}t$ *captures the probability of the activity* a *being already completed at time* t*. For activity* a *with the normal probability density function* $f_a(t)$*, the cumulative probability function is*

$$F_a(t) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{(t-\mu)^2}{2\sigma^2}} \mathrm{d}t \tag{2}$$

Note, that the used activity lifecycle allows the distinction between preparation time (init to enabled)*, idle time* (enabled to running)*, and execution time* (running to terminated) *of an activity. For simplicity reasons, and without restricting the applicability of the approach, we further do not distinguish between these three states and use the term activity duration for the sum of preparation time, idle time and execution time.*

The distributions of the activity durations have to be assessed initially by medical experts, which tend to outperform automatically estimated durations [14], and can then be aligned to historical data to better reflect reality. However that alignment is out of scope of this paper and we will address that in future work. Further, we assume that the distributions given in the model reflect reality.

We assume, that activity durations are independent and can be simulated by independent random variables with given probability distributions δ_i.

Thus the probability density function $S_i(t)$ of the sum of several consecutive sequential activities' durations $\delta_i(t)$ can be calculated by applying the *convolution operation* (cf. equation 3) successively on their probability density functions:

$$S_i(t) = \begin{cases} \delta_i(t), & \text{if } i = 0 \\ \int\limits_{-\infty}^{\infty} S_{i-1}(u)\, \delta_i(t-u)\, \mathrm{d}u, & \text{if } i > 0 \end{cases} \tag{3}$$

We assume that probabilities are assigned to each outgoing edge of a splitting XOR-gateway. Estimation of these probabilities can be done by experts or also mined from execution logs [15]. Further, we assume faithful execution of the process models, a restriction that we aim to get rid of in future work.

3 Related Work

This paper is located in the discipline of business process intelligence [16], where techniques of analysis, prediction and monitoring play a key role.

3.1 Statistical Process Analysis

Some early statistical analysis techniques for graph based models were developed for project planning purposes, e.g., critical path method [17] or PERT [18]. Martin presented an algorithm to compute the total time in a directed acyclic network with times based on independent random variables [19]. He described the convolution operation algorithm for piecewise polynomial distribution functions. However, he did neither address exclusive parts in the network, nor updating mechanisms for different places in the net.

A successor of PERT is called GERT and supports a quite detailed modeling spectrum [20] including statistical estimations of task durations with deterministic or probabilistic edges. The difference to the approach presented in this paper is that we deal with continuous updates of the model where parts are not monitored. That means, we can do more precise estimations of what has happened by looking backwards (c.f. Section 4.4).

3.2 Simulation Based Systems

De Vin et al. show how simulation models can be used for decision support for virtual manufacturing, when enriched with historical data and snapshot data [21]. The line of thought is similar and the authors stress that an integration of different data sources in a common information center is necessary. However, they do not provide a concrete framework, of how to achieve the ideas, but only sketch possible use cases and outline a necessary architecture with the focus on information fusion [22].

3.3 Process Mining

If no workflow system supports the execution of a process, information can still be derived from enterprise resource planning systems, hospital information systems and the like. The term *process mining* specifies the extraction of information, e.g., process models, from event logs that typically exist in these systems [23,24].

Besides extraction, process mining can also be used as conformance checking method [25,3] and extension of existing models with additional (mined) information like branching probabilities [15] or duration of activities [26,9]. In their work [9], van der Aalst et al. describe a system that learns remaining time durations from logs and enriches the traces in the log with the remaining duration. They assume normally distributed times for activities, but they do not consider activities that are not in the log, but only in the model. They describe an algorithm that abstracts from single activity durations and focus on remaining cycle time. In this paper we describe a method to calculate finer grained estimations of probabilities of single activities being active in one process instance.

3.4 Other Monitoring Approaches

In the integrated European project SUPER[1], an approach to monitoring based on domain specific languages was proposed by Gonzáles [27]. In his PhD thesis, he designed an architecture and domain specific language for monitoring processes. However, he does not deal with missing information, that can be probabilistically estimated. Also from the same project is the work of Pedrinaci et al. [28], where semantic concepts are introduced into business process monitoring. However their focus is on ontologies and they use them to integrate different sources of monitoring [28], and align and evaluate goals [29]. They also considered the combination with process mining in [30], where the challenges for semantic enrichment of process mining are outlined.

[1] `http://www.ip-super.org`

In a white paper, DeFee and Harmon presented an idea to integrate simulation and business activity monitoring [31], however, they used discrete event simulation and did not address the use case of missing monitoring points in the model.

In the same vein, Kang et al. [32] recently proposed a probabilistic business process monitoring system that updates its estimates upon arrival of a new event. We also advocate the idea to use a probabilistic approach to monitoring. However, their focus is on estimating failure probabilities of instances and they use support vector machines to achieve that, while in this paper we propose a finer grained prediction for activity instance durations and their lifecycle states.

4 Monitoring with Missing Monitoring Points

In this section, we address the question raised in Sect. 1: *Which tasks in the clinical process are probably being executed at a given time?*

To answer this question for a particular process with process model P, we need access to the occurring events that are correlated with the model through monitoring points. For a particular instance $i \in \mathcal{I}$ of P, we require at least one event $e \in \mathcal{E}$ with $prop_I(e) = i$. Further, there has to be a monitoring point defined for P that maps to the event type of e, i.e. $\exists\, a \in A, \exists\, t \in T : m((a,t)) = prop_{EType}(e)$.

The idea is to calculate the probability distribution functions for each possible monitoring point after activity a's state transition t, i.e., each following activity lifecycle state transition. Since we assume independence of the monitored events, we can use the convolution operation on the distributions of two successive events, to derive the distribution of the combined event of both events having happened.

4.1 Monitoring Sequences with Missing Monitoring Points

We demonstrate the approach with normally distributed random variables for predicting time durations for activities in the model. Thus the computation of the probability density function of the sum of two sequential activities $a_1, a_2 \in A$ with $\delta(a_1) = \mathcal{N}(\mu_1, \sigma_1^2)$ and $\delta(a_2) = \mathcal{N}(\mu_2, \sigma_2^2)$ can be done with the following formula, since normal distributions are closed under convolution:

$$f(a_1 + a_2) = \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2) \tag{4}$$

With the density function given, the cumulative distribution function can be derived through integration of the density function. The cumulative distribution function captures the probability $P(E)$, that an event E, e.g., the termination of an activity, has already happened at a given time. The complementary event $\bar{E}$ of event E captures that the termination of the activity has not happened yet, i.e., that the activity is *still active*.

$P(\bar{E})$ can be calculated by $P(\bar{E}) = 1 - P(E)$. Let two consecutive points of interest p_1 and p_2 ($p_2 > p_1$) have corresponding cumulative distribution functions F_1, F_2. We get the probability of being in the state between these two points of interest by subtracting the two probability distributions of $P(\bar{E}_2) - P(\bar{E}_1)$ at given point in time.

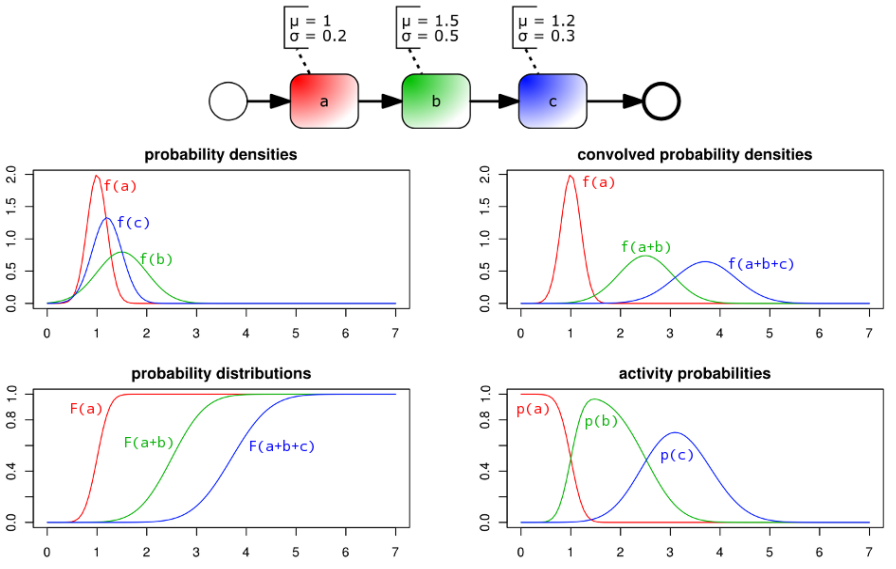


Fig. 4. Process model on top shows sequence of three activities a, b and c with duration parameters of the normal distribution (μ and σ). From top left to bottom right: 1) density functions of the activities, 2) convolved density functions, 3) distributions of end events, 4) probabilities, that activities are being performed at time t.

Fig. 4 visualizes the idea in a simple sequential example. The last graph shows the subtracted cumulative probability functions indicating the probabilities $p(a)$ of activity a currently running (red curve), $p(b)$ (green curve) and $p(c)$ (blue curve) as functions over time t. We can use these functions to predict the probabilities of executing the corresponding activities at time t.

With the calculation of probability distributions defined for sequences, we turn to the remaining control flow constructs in the next subsections.

4.2 Monitoring Choices with Missing Monitoring Points

We assume, that for each outgoing edge of each branching exclusive gateway $g \in G_{XOR}$ in process model P a probability is given that indicates which fraction of cases follows that direction. We introduce a branch probability factor $\gamma \in [0, 1]$ that scales the probabilities in a branch after a split. For instance, in Fig. 5 the process model contains two branches after activity $a1$ with branching factors $\alpha = 0.4$ for the upper branch and $\beta = 0.6$ for the lower branch. Computation of the probabilities of $a2$ and $a3$ are similar to the computation of activities in sequence, with the difference, that the scaling factor γ sets an upper limit to the probabilities. In the example in Fig. 5, the probability of activity b, can be computed by the formula $p(b) = \gamma \cdot (F(a) - F(a+b))$ with $\gamma = \beta = 0.4$ To calculate the branch probabilities of the outgoing branches of an XOR-Gateway, one has to multiply the initial γ branch probability with the probability of the outgoing edge.

At the joining counterpart, the incoming branch probabilities are summed up again to get the probability of the outgoing branch. The probability function of the joining XOR-gateway is computed by the weighted sum of the preceding probability functions on the incoming edges, where the weights are the branch probability factors.

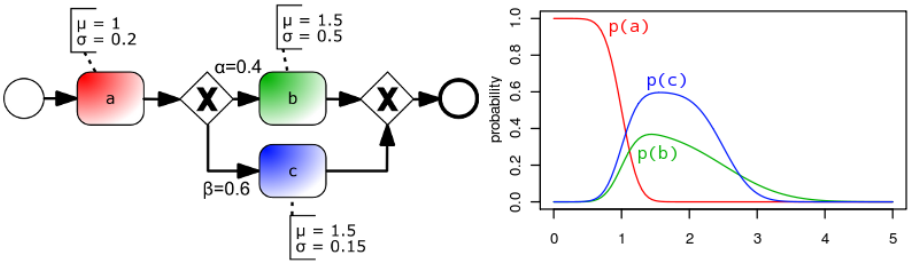


Fig. 5. A choice construct and their respective probabilities

4.3 Monitoring Parallelism with Uncertainty

The calculation of the probabilities of parallel tasks is straight forward. The procedure is the same as for calculating activities in sequence. We just have do that for each outgoing branch after a parallel split $g \in G_{AND}$. The branch probability γ of the incoming branch of the parallel split g is assigned to each of the outgoing branches.

When synchronizing parallel branches, the assumed soundness of the monitored process model assures, that the joined branch probabilities $\gamma_1, \ldots, \gamma_n$ on the synchronized n branches are equal. Otherwise the process would contain a potential deadlock. Thus, the branch probability of the outgoing edge is equal to the incoming ones.

The outgoing *cumulative* distribution function at the synchronization point is the product of the incoming *cumulative* distribution functions, since both branches need to finish before continuing after the joining gateway. Thus to get the probability distribution function, an implementation needs to integrate the probability distribution functions, multiply the integrals and differentiate the result.

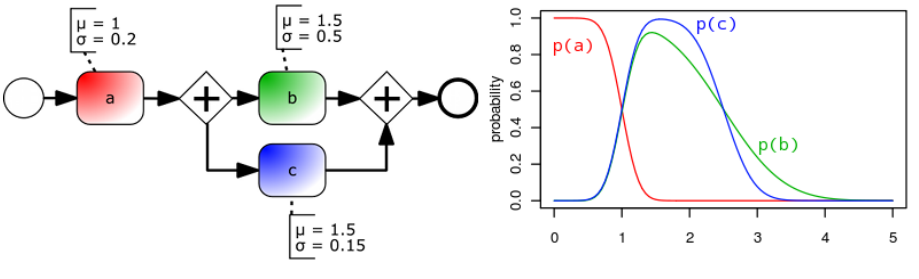
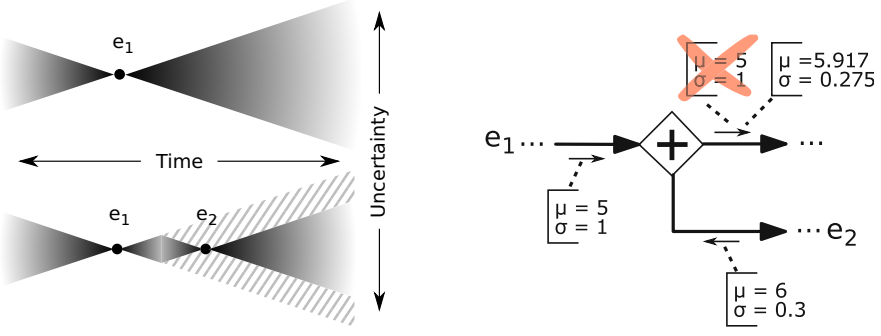


Fig. 6. A parallel construct with three activities and their respective probabilities

4.4 Uncertainty Update

Since the application of the convolution function always adds uncertainty to the result of two random variables, i.e., the squared standard deviations are added, the further the model looks ahead, the more uncertain the estimations become. The upper part of



(a) Uncertainty over time upon arrival of events e_1 and e_2

(b) once second estimation from e_2 is available, the two estimations are combined to get a more accurate estimation for the upper branch.

Fig. 7. Uncertainty update upon arrival of second event. The striped uncertainty area can be eliminated by estimating in both directions from e_2.

Fig. 7(a) shows this graphically. It depicts the uncertainty starting from one monitored event e_1. If, at a later point in time, a second event is detected for a monitoring point, we can update uncertainties in both directions from e_2. An update of the probability distributions in a backward direction becomes useful, if there are parallel branches starting between the two monitoring points, as depicted in Fig. 7(b). The new parameters of the merged estimation can be calculated by multiplying the two normal distributions and rescaling the product to have an area of one. The scaled product of two normal distributions is again a normal distribution with the following parameters:

$$\sigma_{1,2} = \sqrt{\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} \text{ and } \mu_{1,2} = \frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2} \tag{5}$$

5 Applicability Evaluation

We have made two assumptions regarding the activity durations. We assumed, that they are independent and that they can be characterized by a distribution function, i.e., they follow a distribution function instead of being completely random. In the following, we show, that these assumptions hold in a real world use case.

For the motivating use case from Sect. 1, the liver transplant surgery in the University Hospital of Jena, detailed process models exist. Fig. 8 shows a part of the surgery process with some monitoring points marked with ovals. The surgery process has been modeled in a detailed way for various reasons, including education of young doctors. However, for the monitoring, the details of the process model are not relevant and the modeling capabilities defined for this paper suffice.

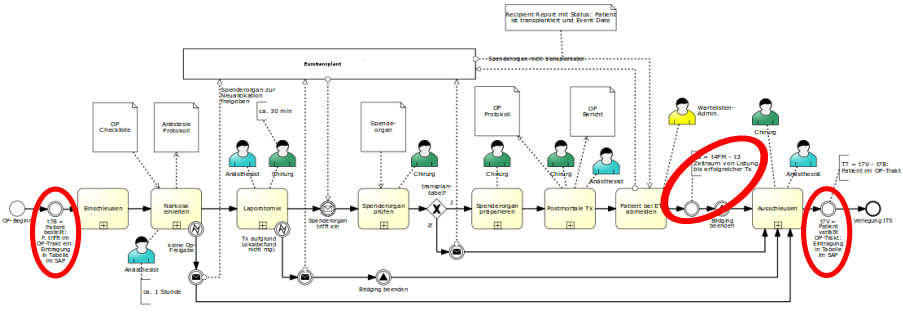


Fig. 8. Part of the liver transplant surgery model with 3 monitoring points (marked with red ovals) modeled in BPMN as annotated Events

We have access to protocol data of surgeries performed from 2009 to 2011. A total of 187 complete cases, each with 11 time stamps for specific events in the surgery, were tracked. The data fits the process models, i.e., can be correlated with activities.

To test for independence, we calculated the correlation values for the durations between these time stamps. The results were low correlation values with a mean of 0.08 and and maximum of 0.24. This indicates independence of activity durations.

In order to analyse the data for characteristic distributions, we have statistically investigated the durations between successive events. Figure 9 shows the duration from begin of the anesthesia procedure of the patient to the approval of the anesthesia team for surgery. The normal distribution fits the durations quite nicely, as one can see in the linear alignment of the QQ-plot on the right.

Literature characterizes the duration of surgeries in hospitals as log-normally distributed [12] with a skewed right tail for cases that take longer due to complications. To test the data for this claim, we used a maximum likelihood estimation for our sample durations from “cut” to “suturation”. And although we have only a small sample, the p-value of a fitted log-normal function to the duration between cut and suturation yields p=0.157 for the Cramer-von-Mises test, which is a strong indicator that the data is indeed log-normally distributed. The histogram of the data is shown in Fig. 10 with the estimated distribution and the QQ-Plot. The data contains a a second heap at around 8-9 hours as seen in the histogram in Fig. 10.

From analysis of this data, we can argue, that the assumptions were reasonable, though further analysis with greater sample sizes and more data would be necessary to bolster them.

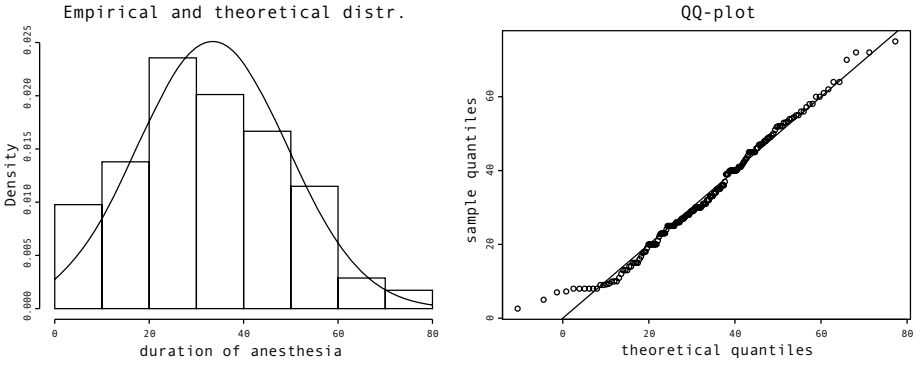


Fig. 9. Duration from "anesthesia begin" to "anesthesia approval", fit with normal distribution

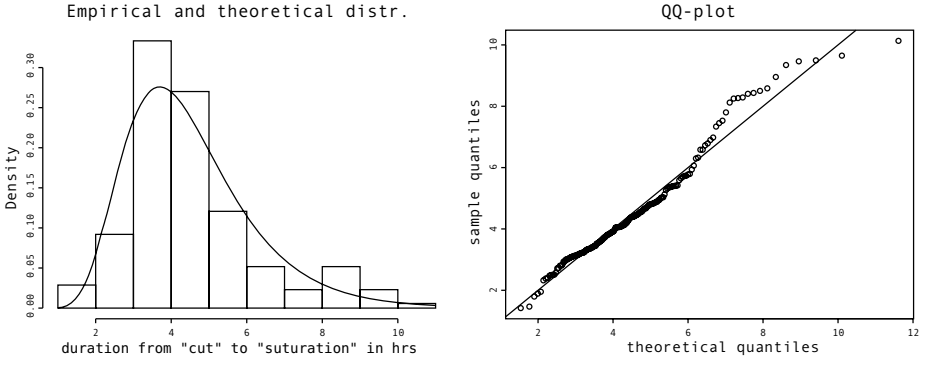


Fig. 10. Duration from "cut" to "suturation", fit with log-normal distribution

6 Conclusion

In this paper we have investigated the scenario of monitoring a process that is executed without a workflow engine, but described by a detailed process model, which we encountered in a hospital setting. To answer the question, which activities are probably being executed at a given time, we presented a method to calculate probabilities for a core set of modeling capabilities and discussed a special case, when backward estimation yields improved monitoring results for parallel branches.

By applying the proposed techniques in a model where scattered monitoring points exist, relevant and interesting monitoring questions can be answered probabilistically and the estimations get better, the more events are detected at monitoring points.

We evaluated the applicability of our approach by analyzing real durations from a liver transplant surgery and investigated, whether this data fits into characteristic probability distributions and is stochastically independent.

One limitation of the approach is, that we assume faithful execution of the process models, which might not hold in health care scenarios, where exceptions are common [4]. We limited the allowed modeling constructs to a minimum and disregarded loops in the model, but we want to address these limitations in future work. Also the quality of the predictions in unmeasured branches heavily depends on the estimations provided by experts. However, these can be improved by real measurements and alignment to the durations between the monitoring points.

The work presented here is motivated by a hospital scenario, but is is applicable to other domains, too. It is applicable, when information on process progress can be made available as events in an IT-infrastructure and when detailed process models exist that describe truthfully what actually happens. The gain in process transparency should always justify the additional costs of implementing this approach.

References

1. McCoy, D.W.: Business Activity Monitoring: Calm Before the Storm. Gartner Research Note LE-15-9727, vol. 15 (2002)
2. Dahanayake, A., Welke, R.J., Cavalheiro, G.: Improving the Understanding of BAM Technology for Real-Time Decision Support. International Journal of Business Information Systems 7, 1–26 (2011)
3. Rozinat, A., van der Aalst, W.M.P.: Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems 33, 64–95 (2008)
4. Lenz, R., Reichert, M.: IT Support for Healthcare Processes Premises, Challenges, Perspectives. Data & Knowledge Engineering 61, 39–58 (2007)
5. van der Aalst, W.M.P.: Verification of Workflow Nets. In: Application and Theory of Petri Nets 1997, pp. 407–426 (1997)
6. zur Muehlen, M., Recker, J.: How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 465–479. Springer, Heidelberg (2008)
7. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow Resource Patterns: Identification, Representation and Tool Support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)
8. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer-Verlag New York Inc. (2007)
9. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time Prediction Based on Process Mining. Information Systems 36, 450–475 (2011)
10. Hollingsworth, D., et al.: Workflow Management Coalition: The Workflow Reference Model. Workflow Management Coalition (1993)
11. Liu, X., Chen, J., Yang, Y.: A Probabilistic Strategy for Setting Temporal Constraints in Scientific Workflows. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 180–195. Springer, Heidelberg (2008)
12. Strum, D.P., May, J.H., Vargas, L.G.: Modeling the uncertainty of surgical procedure times: comparison of log-normal and normal models. Anesthesiology 92, 1160–1167 (2000)
13. Wu, J., Mehta, N., Zhang, J.: A Flexible Lognormal Sum Approximation Method. In: GLOBECOM 2005, pp. 3413–3417 (2005)
14. Wright, I.H., Kooperberg, C., Bonar, B.A., Bashein, G.: Statistical Modeling to Predict Elective Surgery Time – Comparison with a Computer Scheduling System and Surgeon-Provided Estimates (1996)

15. Rozinat, A., van der Aalst, W.M.P.: Decision Mining in ProM. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 420–425. Springer, Heidelberg (2006)
16. Grigori, D., et al.: Business Process Intelligence. Computers in Industry 53, 321–343 (2004)
17. Kelley Jr., J.E., Walker, M.R.: Critical-Path Planning and Scheduling. Papers Presented at the Eastern Joint IRE-AIEE-ACM Computer Conference, December 1-3, pp. 160–173. ACM (1959)
18. Moder, J.J., Phillips, C.R.: Project Management with CPM and PERT (1964)
19. Martin, J.J.: Distribution of the Time through a Directed, Acyclic Network. Operations Research 13, 46–66 (1965)
20. Neumann, K.: Recent Advances in Temporal Analysis of GERT Networks. Zeitschrift für Operations Research 23, 153–177 (1979)
21. de Vin, L.J., Ng, A.H.C., Oscarsson, J.: Simulation-based Decision Support for Manufacturing System Life Cycle Management. Journal of Advanced Manufacturing Systems 3, 115–128 (2004)
22. de Vin, L.J., Ng, A.H.C., Oscarsson, J., Andler, S.F.: Information Fusion for Simulation based Decision Support in Manufacturing. Robotics and Computer-Integrated Manufacturing 22, 429–436 (2006)
23. Agrawal, R., Gunopulos, D., Leymann, F.: Mining Process Models from Workflow Logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
24. Weijters, A., van der Aalst, W.M.P.: Process Mining: Discovering Workflow Models from Event-Based Data. In: BNAIC 2001, Citeseer, pp. 283–290 (2001)
25. van der Aalst, W.M.P.: Business Alignment: Using Process Mining as a Tool for Delta Analysis and Conformance Testing. Requirements Engineering Journal 10, 198–211 (2005)
26. van Dongen, B.F., Crooy, R.A., van der Aalst, W.M.P.: Cycle Time Prediction: When Will This Case Finally Be Finished? In: Meersman, R., Tari, Z. (eds.) OTM 2008. LNCS, vol. 5331, pp. 319–336. Springer, Heidelberg (2008)
27. González, O.: Monitoring and Analysis of Workflow Applications: A Domain-specific Language Approach Ph.D. thesis, Universidad de los Andes (2010)
28. Pedrinaci, C., et al.: SENTINEL: A Semantic Business Process Monitoring Tool. In: Proceedings of the First International Workshop on Ontology-Supported Business Intelligence, pp. 1–12. ACM (2008)
29. Pedrinaci, C., Markovic, I., Hasibether, F.: Strategy-Driven Business Process Analysis. In: Business Information, pp. 1–12 (2009)
30. Alves de Medeiros, A.K., Pedrinaci, C., van der Aalst, W.M.P., Domingue, J., Song, M., Rozinat, A., Norton, B., Cabral, L.: An Outlook on Semantic Business Process Mining and Monitoring. In: Meersman, R., Tari, Z. (eds.) OTM 2007 Ws, Part II. LNCS, vol. 4806, pp. 1244–1255. Springer, Heidelberg (2007)
31. DeFee, J.M., Harmon, P.: Business Activity Monitoring and Simulation. BP Trends Newsletter, White Paper and Technical Briefs, pp. 1–24 (2004)
32. Kang, B., Kim, D., Kang, S.: Periodic Performance Prediction for Real-time Business Process Monitoring. Industrial Management & Data (2011)