Temporal Anomaly Detection in Business Processes*

Andreas Rogge-Solti[1] and Gjergji Kasneci[2]

[1] Vienna University of Economics and Business, Austria
andreas.rogge-solti@wu.ac.at
[2] Hasso Plattner Institute, University of Potsdam, Germany
gjergji.kasneci@hpi.uni-potsdam.de

Abstract. The analysis of business processes is often challenging not only because of intricate dependencies between process activities but also because of various sources of faults within the activities. The automated detection of potential business process anomalies could immensely help business analysts and other process participants detect and understand the causes of process errors.
This work focuses on temporal anomalies, i.e., anomalies concerning the runtime of activities within a process. To detect such anomalies, we propose a Bayesian model that can be automatically inferred form the Petri net representation of a business process. Probabilistic inference on the above model allows the detection of non-obvious and interdependent temporal anomalies.

Keywords: outlier detection, documentation, statistical method, Bayesian networks

1 Introduction

Business process management is the key to aligning a company's business with the needs of clients. It aims at continuously improving business processes and enabling companies to act more effectively and efficiently. The optimization of business processes often reveals opportunities for technological integration and innovation [21]. Despite these positive aspects, business processes are often complex by containing intricate dependencies between business activities. Moreover, the activities are enacted in a distributed fashion and in environments where faults can occur [22]. Thus the analysis of business processes is a highly challenging task [11], even for experts.

Automated mining of process patterns out of event data can reveal important insights into business processes [2]. However, the performance of process mining algorithms is highly dependent on the quality of event logs [3], which in turn are also crucial for documentation purposes [17]. Most data mining algorithms build on the unrealistic assumption that the recorded training data is valid and representative of the data expected to be encountered in the future. In process mining such an assumption makes sense only if documentation correctness is guaranteed. Most work on documentation correctness deals with only structural aspects, i.e., with the *order* of execution of activities [18,4,7].

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

This work proposes a novel approach to detect temporal outliers in activity durations. Outliers can have various causes; they can be obvious, e.g., in case of non-typical measurement or execution errors [10], and they can be hidden, e.g., in case of latent or propagated errors that do not reveal themselves as such during the execution. Often, however, it is sufficient to detect potential anomalies and not the exact errors. When presented with such anomalies, expert analysts or other process participants can dig deeper into the problem and fix any present error. Hence, detecting potential anomalies can immensely simplify the task of finding potential errors in business processes [5].

The focus of this work is on temporal anomalies, where a group of interdependent activities has a non-typical runtime or delay pattern. Note that this is different from the detection of delay for a single activity, as a group of activities may still show a regular overall runtime, even if the single activities have anomalous delays. Hence we go beyond the detection of delay for single activities and are generally interested in implausible delay patterns within a group of activities. Our goal is to detect such anomalies from event traces and to extrapolate from the investigation of delay for single activities to the detection of temporal anomalies in the entire case.

The main achievements of this work are:

- An extensive analysis of general properties of temporal anomalies based on event logs and the formalism of stochastic Petri Nets
- A principled formalization of temporal anomalies based on approximate distributions of activity durations
- A probabilistic approach for reliably detecting temporal anomalies in sequences of consecutive activities by analyzing the corresponding event logs
- An extensive evaluation of the approach based on synthetic as well as real-world datasets with labeled error occurrences.

The remainder of the paper is organized as follows. Section 2 introduces the main concepts this work builds on. Section 3 gives an overview of related work and sets the context for the achievements presented in this paper. The approach for temporal anomaly detection is presented in Section 4. An extensive experimental evaluation of the approach is presented in Section 5, before concluding in Section 6.

2 Preliminaries

Understanding the business processes of an organization can be facilitated by business process models. We assume that a business process model is available and accurately describes the behavior of a process. There exist many competing modeling languages for business processes, of which we abstract by relying on the Petri net [15] representation of the models [12], which are able to capture the most important workflow-patterns used in different languages. We define Petri nets according to the original definition [15] as follows.

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

- P *is a set of places.*
- T *is a set of transitions.*
- $F \subseteq (P \times T) \cup (T \times P)$ *is a set of connecting arcs representing flow relations.*

We also define paths in Petri nets as follows. Let F^+ denote the transitive closure over F, then a *path* exists between any two nodes $l, n \in (P \cup T)$, iff $(l, n) \in F^+$. We assume that the models are *sound* workflow nets [1], i.e., that they have a dedicated start place p_i and an end place p_o, each node lies on a path between p_i and p_o, there are no deadlocks, and whenever a marking with $p_o > 0$ is reached, all other places $p \in \{P \setminus p_o\}$ are empty, i.e., the process is properly terminated. We do not put further restrictions on the supported model class and explicitly also support non-structured and non-free-choice parts in the models.

During execution of single instances of business processes, information regarding the state and progress of these process instances are recorded in various information systems. For example, a logistics service provider tracks the position of its transport means, or a financial institute tracks the status of customer loan requests. We assume that the progress information for each case is available and collected in event logs [2]. Therefore, we use the established notion of event logs that contain executed traces.

Definition 2 (Event Log). *An event log over a set of activities A and time domain TD is defined as* $L_{A,TD} = (E, C, \alpha, \beta, \gamma, \geq)$*, where:*

- E *is a finite set of events.*
- C *is a finite set of cases (process instances).*
- $\alpha : E \rightarrow A$ *is a function assigning each event to an activity.*
- $\beta : E \rightarrow C$ *is a surjective function assigning each event to a case.*
- $\gamma : E \rightarrow TD$ *is a function assigning each event to a timestamp.*
- $\geq \subseteq E \times E$ *is the succession relation, which imposes a total ordering on the events in E.*

We require that the time of occurrence is recorded for event entries in an event log. For example, in a hospital, where nurses record the timestamps of certain treatment steps in a spreadsheet, we can derive an event log of that spreadsheet, where each case is associated to a row in the spreadsheet and each activity corresponds to a column.

In previous work, we have provided an algorithm to enrich *PN* models with statistical execution information in event logs [16]. The statistical information that we learn from historical executions is associated with transitions that capture decision probabilities, and with transitions that capture process activities and their corresponding durations. We revisit the definition of the enriched stochastic model [16].

Definition 3 (Generally Distributed Transition Stochastic Petri Net). *A generally distributed transition stochastic Petri net (GDT_SPN) is a tuple:*
$GDT\_SPN = (P, T, \mathcal{W}, F, \mathcal{D})$*, where* (P, T, F) *is the basic underlying Petri net. Additionally:*

- *The set of transitions* $T = T_i \cup T_t$ *is partitioned into immediate transitions* T_i *and timed transitions* T_t*.*
- $\mathcal{W} : T_i \rightarrow \mathbb{R}^+$ *assigns probabilistic weights to the immediate transitions.*
- $\mathcal{D} : T_t \rightarrow D$ *is an assignment of arbitrary probability density functions D to timed transitions, reflecting the duration of each corresponding activity.*

Probability density functions represent the relative number of occurrences of observations in a continuous domain. In most real business processes, analytical expressions

for probability density functions will not be available, and we resort to *density estimation* techniques [19]. For example, kernel density estimation techniques as described by Parzen [13] are a popular method to approximate the real distribution of values.

Once we extracted the stochastic properties of past executions, we can check whether new traces match the *regular* and *expected* behavior, or if they are *outliers* and deviate from the stochastic model. We are interested in finding temporal anomalies to assist business analysts in root-cause analysis of outliers. Further, we aim at separation of outliers that can occur and are expected during execution from *measurement errors*, e.g., when in the above example a nurse enters a wrong time for an activity by mistake.

The idea of this paper is to exploit knowledge that is encoded in the process model for this task. The problem that we encounter, if we only have an event log that traces the execution of activities, is that it is not made explicit, which activities are dependent on which predecessors, because of possible parallel execution and interleaving events. Therefore, we extract structural information from the *GDT_SPN* model—in fact, the underlying *PN* model already contains this information.

To ease the discussion of temporal outlier detection in the main section, we introduce the concepts of control flow and temporal dependence.

Definition 4 (Control-flow Dependence). *Let $t_1, t_2 \in T$ be two transitions. A* control-flow dependence *exists between t_1 and t_2, iff there is a path between t_1 and t_2.*

We further define *temporal dependencies* on a process instance level. That is, we want to identify the transitions that are immediately enabled after the current activity finished. We therefore replay the cases from the event log, as proposed in [18,4], and additionally keep track of the global clock during replay [16].

Definition 5 (Temporal Dependence, Direct Dependence). *Let $t_1, t_2 \in T_t$ be two transitions of a GDT_SPN model. There is a* temporal dependence *in a case between t_1 and t_2, iff the timestamp of termination of t_1 is equal to the global clock, when t_2 becomes enabled. That is, there is no other timed transition firing between the firing of t_1, and the enabling of t_2.*

A direct dependence *between t_1, and t_2 exists, iff there is a temporal dependence between t_1, and t_2 and there is a control-flow dependence between t_1, and t_2.*

For notational convenience, we define the dependence relation $dep : A \times A$ on the corresponding process activities of the model that contains all pairs of activities, of which the timed transitions in the model are in a *direct dependence* in a case. This simple definition of *dep* works well for process models without loop constructs. If there exist loops in a process model, there could be a *direct dependence* between a transition and itself (in the next iteration), and likewise the corresponding activity would be in a self-dependence. Therefore, we will use the dependence relationship dep_a on individual instances of activities, instead of on the activity model. It is straightforward to see that we can enumerate multiple instances of the same activity and thus limit the direct dependencies of an activity instance to the activity instances that follow directly. Note that the number of the set of activities that directly depends on an activity is in most cases 1, unless there is a parallel split after the activity. In the latter case, the number of directly dependent activities equals the number of parallel branches in the process

that are triggered with the termination of the activity. Also note that if we restrict our attention to activity instances in executed process instances, an activity instance followed by an exclusive choice between several alternative branches in the control flow still only has 1 activity with which it is in a direct dependence. Latter is the first activity on the path that was chosen.

Given different instances of activities and the dependence relation dep_a, one can represent their duration dependencies by means of a Bayesian network. In such a network, nodes represent duration variables that follow an estimated prior distribution that is encoded in the *GDT_SPN* model, or they represent points of occurrence of events, such as the termination of an activity. It is straight-forward to see that such a network can be directly derived from an instantiation of the activity model, which in turn can be created during replay of each case of the event log. Later on, we will show that because of the simple structure of this Bayesian network, effectively we only need a window-based analysis of the events and their direct dependencies in the event log.

Definition 6 (Bayesian Network). *Let $\{X_1, \ldots, X_k\}$ be a set of random variables. A Bayesian network BN is a directed acyclic graph (N, F), where*

- *$N = \{n_1, \ldots, n_k\}$ is the set of nodes assigned each to a random variable $X_1, \ldots, X_k$.*
- *$F \subset N \times N$ is the set of directed edges.*

Let $(n_i, n_j) \in F$ be an edge from parent node n_i to child node n_j. The edge reflects a conditional dependence between the corresponding random variables X_i and X_j.

Each random variable is independent from its predecessors given the values of its parents. Let π_i denote the set of parents of X_i. A Bayesian network is fully defined by the probability distributions of the nodes n_i as $P(X_i \mid \pi_i)$ and the conditional dependence relations encoded in the graph. Then, the joint probability distribution of the whole network factorizes according to the chain rule as $P(X_1, \ldots, X_N) = \prod_{i=1}^{N} P(X_i \mid \pi_i)$.

Although nodes are most commonly used for capturing random behavior, it is not difficult to introduce deterministic nodes in a Bayesian network. A node in a Bayesian network can be assigned a single value with probability 1 or a (deterministic) function of the values of the parent nodes. For the purposes of this paper, we limit our attention to the most common workflow structures: *sequence*, *exclusive choice*, *parallelization* and *synchronization* of control flow. Because exclusive choices are removed during execution, two deterministic constructs are sufficient to capture the dependencies between durations of activity instances and timestamps of events: we need the *sum* operation to capture sequential dependencies and the *max* operation to model the synchronization of parallel activities. Latter two nodes deterministically assign probability 1 to the sum (resp. max) of the parent variables' values and probability zero to other values.

An example model from a hospital process shall serve to illustrate this point. Figure 1 shows a scenario, where a patient arrives at the operating room (t_A) and is then treated with antibiotics (t_B), while the induction of anaesthesia (t_C) is conducted in parallel. After both these activities are completed, the surgery t_D can be performed. In this example, depending on the current case, the direct dependence relation is $dep_a = \{(a, b), (a, c), (b, d)\}$, if activity b takes longer than c, or $dep_a = \{(a, b), (a, c), (c, d)\}$ in the other case. Note that the faster of the parallel activities has no temporal dependence to activity d, and is thus not included in the dependence relation.

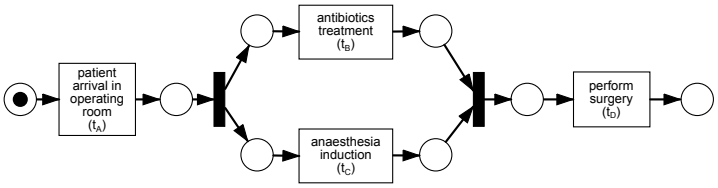
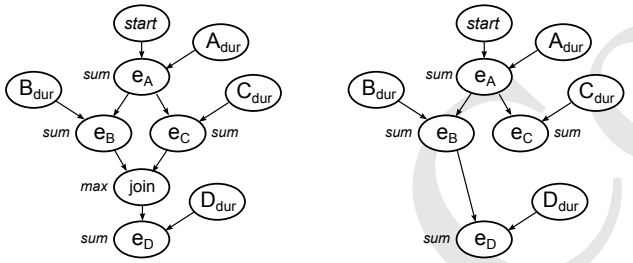


Fig. 1: Surgery example *GDT_SPN* model depicting dependencies between activity durations in a process model with sequence and parallel split and merge constructs.



(a) Bayesian model. (b) Simplified Bayesian model.

Fig. 2: Bayesian network models for the surgery example depicted in Figure 1. If we know that e_B happens after e_C in case (b), we can simplify the dependencies by removing the max node and the dependence from e_C to e_D, and only maintain the dependence between e_B and e_D.

The resulting Bayesian model is shown in Figure 2a. The process is started at a certain point in time, which can be aligned to zero in the model. The duration of activity a is modeled as A_{dur}, and influences the value of e_A that captures the observed timestamp of the corresponding termination event. Latter is the last activity before the control flow is split into two parallel branches. For each, we add the corresponding activity duration to the resulting events. Then, the maximum of the parallel branches is selected by the deterministic max node *join* and the final event e_D is the sum of latter and the duration of activity d captured in D_{dur}.

Given the dependency relation dep_a, we can further simplify the Bayesian network, as shown in Figure 2b. In this example, the max node is resolved due to the knowledge that the branch with activity b finished after the branch with activity c. The dependency from e_C to e_D can be removed as well. Note that in general it is not always the case that there will be a single transition in the model for a process activity. If there is more information available about the activity lifecycle of an activity [21] (e.g., if we know when the activity has been enabled, started, and completed), the model can more accurately capture the different phases. Therefore, an extension by replacing a single timed transitions by a sub-net that captures fine grained activity lifecycle transitions is possible.

3 Related Work

The problem of anomaly detection is to identify data that does not conform to the general behavior or the model of the data. Different flavors of the general problem are also known as outlier analysis, novelty detection, and noise removal. The problem is relevant, because most data that is gathered in real settings is *noisy*, i.e., contains outliers or errors. Various methods (e.g., classification, clustering, statistical approaches, information theoretic approaches) are used for anomaly detection [10]. We refer the interested reader to the survey by Chandola et al. [5] for an overview of different approaches to the problem, and to the text book by Han and Kamber [10] for details on classification and clustering approaches. For this paper, we limit the discussion to statistical methods of outlier detection as well as proposed anomaly detection techniques in the domain of business processes.

Statistical outlier detection is often based on hypothesis tests, that is, on the question whether it is very unlikely to observe a random sample that is *as extreme* as the actual observation [9]. This method can also be applied in combination with non-parametric methods, as proposed by Yeung and Chow [23]. They use a non-parametric approach and sample from the likelihood distribution of a kernel density estimate to check whether new data is from the same distribution. We shall describe the approach by Yeung and Chow in more detail in the main section, as our work builds on the same idea.

Much attention has been devoted to the detection of structural anomalies in process event logs [18,4,7], which affect the performance of process mining algorithms [2]. Sometimes, these anomalies are considered harmful, i.e., if they violate compliance rules [8]. Techniques range from algorithmic replay [18], to cost-based fitness analysis [4] that is able to guarantee to find an optimal solution to the alignment of model and log based on distinct costs for not synchronized parts.

Cook et al. integrate time boundaries into conformance checking by using a formal timed model of a system [6]. They assume that a timed model is already present and specified, and consider time boundaries instead of probability density functions, while we strive to detect anomalies that differ from usual behavior and also want to distinguish measurement errors from regular outliers. Hao et al. analyze business impact of business data on key performance indicators and visualize them either in aggregated or single views in the model [11]. In contrast, our work leverages information encoded in the structure of the model to find related variables and to detect outliers in continuous space.

4 Anomaly Detection in Business Processes

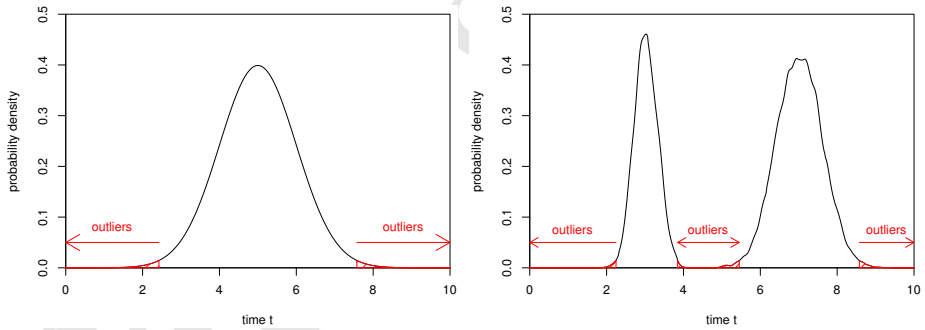
The approach presented in this section builds on intuitive assumptions derived from practical observations. For example, we assume that anomalous event durations are rare when compared to normal execution times. Moreover, we assume that the actual (i.e., typical) activity durations are independent; the observed duration of an activity, on the other hand, depends only on its actual (i.e., typical) duration and the observed duration of the preceding activity. Another important assumption is that the whole process itself is in a so-called "steady state", that is, we do not consider trends or seasonality in the model. Finally, we assume that all events are collected in an event log that contains both

information about the activity as well as the point in time of occurrence. In practice, such logs are maintained for most business process routines. In [16], it was shown that these kinds of logs can be directly used to infer probabilistic models of business processes, for example *GDT_SPN* models. We assume that a *GDT_SPN* model is enriched from a plain Petri net representation of a business process model.

4.1 Detection of Outliers

Most work on detecting deviations from specified control flow only focuses on structural deviations, e.g., whether the activities were performed in the correct order, or whether activities were skipped [18,4,7]. In this work, we would like to go one step further and also consider the execution time of activities to detect cases that do not conform with the expected duration. The latter is encoded in form of statistical information that is encoded in the probability density function of timed transitions in *GDT_SPN*. First, let us recall a simple procedure to detect outliers.

A common rule to find outliers in *normally* distributed data is to compute the z-score of an observation ($z = \frac{x-\mu}{\sigma}$, i.e., the deviation about the mean normalized in units of standard deviations) and classify an observation as an outlier iff $|z| > 3$. This simple method depends on the assumption that the data is normally distributed, which is often not the case.



(a) 1% outliers in a normal distribution.

(b) 1% outliers in a non-parametric kernel density distribution.

Fig. 3: Classification of 1 percent outliers. The areas containing the most unlikely 1 percent are highlighted as outliers.

To gain more flexibility and to not depend on measures in units of standard deviation, it is also possible to specify the threshold for outliers in terms of percentage of the observations. Let us assume we want to find the most extreme 1 percent of a normally distributed observations. Then the we can compute the theoretical 0.5 percent quantile, and the 99.5 percent quantile of the normal distribution and if the observed value falls in the region below the 0.5 percent quantile (or equivalently falls above the 99.5 percent quantile), we classify the observation as an outlier. Figure 3a depicts the region of 1 percent of the most unlikely values in a normal distribution with a mean of 5 and a

standard deviation of 1. Note that this more flexible test that is based on lower and upper quantiles is only valid for *symmetric* probability distributions, like the normal distribution.

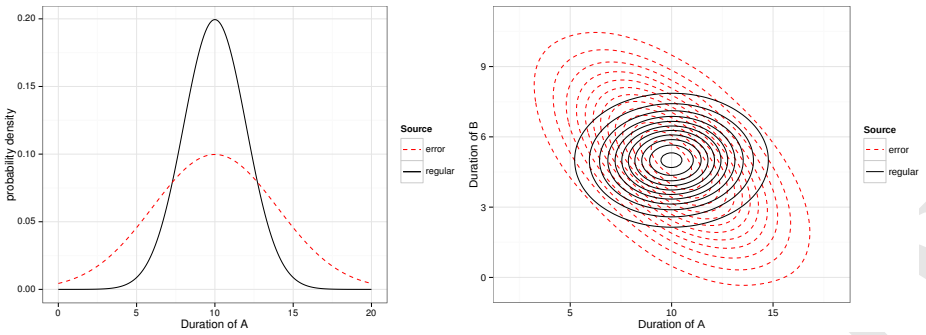
In reality, the assumption of normally distributed values durations is often inappropriate. When dealing with real data, simple parametric models might not be able to capture the probability distributions in sufficient detail. The observations could belong to distinct classes with different behavior (e.g., due to differences in processing speed), which can result in two modes in the probability density function. An example is depicted in Figure 3b, where two peaks in the data are observable at $t = 3$ and $t = 7$. Note that in such cases, the z-value is not suitable any more, and we rely on more robust classification methods, as described by Yeung and Chow [23].

The main idea for the outlier detection is based on a hypothesis test that tries to identify the probability that an observation x is from a particular model $\mathcal{M}$. Therefore, the distribution $L(y) = \log \mathrm{P}(y \mid \mathcal{M})$ of the log-likelihoods of random samples y from the model $\mathcal{M}$ is computed. This can be done by sampling and approximating the probability density function of the log-likelihood of each sample point. The log-likelihood of the event that x was generated by the same model $\mathcal{M}$ is $L(x) = \log \mathrm{P}(x \mid \mathcal{M})$. The hypothesis that needs to be tested is whether $L(x)$ is drawn from the same distribution of log-likelihoods as $L(y)$, i.e., $\mathrm{P}(L(y) \leq L(x)) > \psi$ for a threshold parameter $0 < \psi < 1$. The null hypothesis is rejected if the probability is not greater than ψ, implying that x is an outlier with respect to $\mathcal{M}$.

This method is general and is also applicable for multidimensional data, but we will restrict our focus to the one and two dimensional case. In our approach, this method is the key to identifying temporal anomalies in single activity durations. Depending on the domain, business analysts can use the approach with suitably chosen thresholds according to the expected error rate. Subsequently, on a case-to-case basis, the analysts can browse through the suggested outliers and decide whether they are actual outliers or mere measurement errors. In the following, we present the details of our approach.

4.2 Detection of Measurement Errors

We want to differentiate single measurement errors from benign outliers. Therefore, we exploit the knowledge that measurement errors only affect a *single* event, while an outlier also affects the *succeeding* events. For example, an extraordinary delay in a task is unlikely to be regained immediately by the next task, but rather also cause a delay in the latter. This means that if there is a measurement error in a single activity, usually two activities are affected: the activity, of which the event is describing the completion time, as well as the following activity that is enabled immediately afterwards. We expect that a positive measurement error that indicates a long duration to yield a negative error (too short duration) for the following activity. Figure 4 highlights the difference between single outlier detection and measurement error detection. In Fig. 4a a measurement error can happen for a single activity, where we can only hint at the error by identifying it as an outlier. On the other side, we see two dependent activity durations in Fig. 4b, where we can be more certain that a measurement error occurred, if the data point is better explained by the diagonal error model that shows a high negative correlation between the durations.



(a) duration of a single activity (b) joint duration density of subsequent activities

Fig. 4: Expected behavior (solid black curve) and measurement errors (dotted red curve) in the univariate (single activity) and the bivariate case (two directly depending activities).

A typical assumption when designing an outlier detection model is that outliers will not follow the distribution of the benign data points. However, when dealing with the detection of anomalous sequences of data points, e.g., consecutive durations of activities, the problem is that an anomalous sequence may look benign (e.g., anomalous durations may add up to expected runtimes) until the single data points are analyzed in detail. This problem is related to Simpson's paradox [20], where the signal obtained from an aggregated set of data points coming from at least two different distributions can be corrupted, in the sense that it would hide the signal that one might have obtained by analyzing the data points grouped by their distributions. In [14], Pearl defined a set of specific conditions, i.e., precise criteria for selecting a set of "confounding variables" – that yield correct causal relationships if included in the analysis – to avoid Simpson's paradox. Basically, Pearl advocates the analysis of variable dependencies and their representation by means of Bayesian networks; with probabilistic inference on the networks yielding unbiased signals from the data. We follow this recipe and model a sequence of activity durations as a Bayesian network that is directly derived from the dep_a relation that we introduced in Section 2 for each case.

Basically, our Bayesian network contains activity duration variables (e.g., A_{dur}), and timestamps of events (e.g., e_A), see Section 2.For every pair of dependent activity instances $(a, b) \in dep_a$, we examine their durations A_{dur}, and B_{dur}. Thereby, we reason about the probability of an error having occurred at that pair. More specifically, we compare the bivariate distribution of $\mathrm{P}(A_{dur}, B_{dur} \mid error)$ with $\mathrm{P}(A_{dur}, B_{dur} \mid no\_error)$ and their marginalized versions (including an error), i.e., $\mathrm{P}(A_{dur} \mid error) = \int_{B_{dur}} \mathrm{P}(A_{dur}, B_{dur} \mid error)$, and $\mathrm{P}(B_{dur} \mid error) = \int_{A_{dur}} \mathrm{P}(A_{dur}, B_{dur} \mid error)$, to identify certain error patterns in consecutive events. Specifically, we are interested in the relative likelihood of each of the above conditionals (i.e., relative to the sum of the likelihoods of the four available models). This allows us to select the most plausible model that might have generated A_{dur} and B_{dur}.

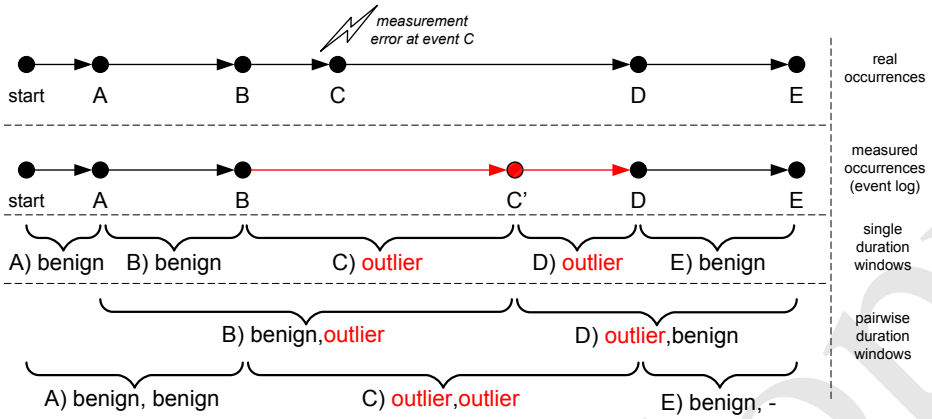


Fig. 5: Window-based measurement error detection approach. First row shows original events, where the timestamp of event C has been corrupted due to an error. By this example error, the duration of C and D are affected. The approach that uses a single duration window has difficulties localizing the error and will usually yield too many false positives. Pairwise comparison of subsequent durations can pinpoint the error location, i.e., if both durations are erroneous and negatively correlated.

The simplicity of the above approach allows us to effectively move a window of size 2 over directly dependent events (which must not be direct neighbors in the log) and analyze the plausibility of their joint runtime as well as their durations in separation. The dependencies gathered from the *GDT_SPN* model are leveraged to find successors of the current event. When the current activity is the last in a dependency chain (i.e., last activity in the process, or last activity in a "fast" parallel branch, where the next activity is not waiting for the fast branch, but for the slow one), we cannot exploit further information to identify outliers as errors. In such cases, we fall back to outlier detection, as described in Section 4.1. If there exists a direct dependency between to the activity and another (i.e., the two activity instances are contained in dep_a), we distinguish 4 cases and probabilistically infer the most plausible one. The four cases are:

benign, benign neither the duration of current event nor the duration of its successor are outliers.

benign, outlier the duration of the successor is an outlier, but not the duration of the current event.

outlier, outlier both durations are outliers and the outliers are strongly correlated negatively. This indicates a measurement error at the current event.

outlier, benign the duration of the current event is an outlier, but not the duration of its successor.

When an activity instance effectively starts multiple activities in parallel (i.e., is contained more than once in dep_a), we simply compute the weighted average of the activity pairs, where by default the weights are distributed evenly. Domain experts could set the weights according to the reliability of single activities error rate.

The most plausible case is decided, based on the likelihood ratios, as described above. Figure 5 gives an overview of this window-based error detection approach. It shows that considering only single activity durations in isolation cannot distinguish between an error of the current activity caused by a local measurement error, or a previous measurement error. The figure also depicts the four cases that we try to distinguish in the pairwise duration window approach.

5 Evaluation

We implemented the anomaly detection mechanism as a plug-in to the process mining framework ProM[3]. Figure 6 shows the graphical user interface of the plug-in. The plug-in allows the user to select a *GDT_SPN* model and an event log to identify the outliers in a case by case fashion. The cases are ordered by the number of outliers per case and by their outlier scores, such that business process analysts ideally only have to scan the top of the list for outliers. In the center of the screen, the model is presented, while the analyst can select individual activities and see the corresponding duration distributions with the current duration marked as a vertical line (top right). Additionally, the log-likelihood distribution of the duration is shown to visually judge the probability that such a value—or a more extreme one—arises assuming that the distribution model in the *GDT_SPN* model is correct.

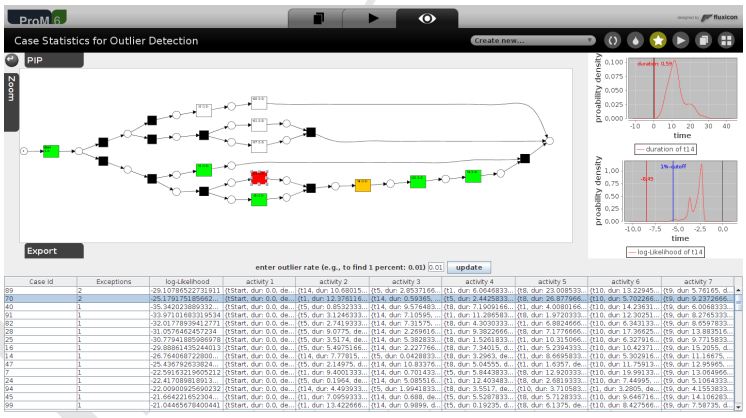


Fig. 6: User interface to the outlier detection plug-in in ProM.

To also evaluate our approach with real data, we analyze the accuracy to detect measurement errors in the event log of a Dutch hospital. We depicted the surgery process model in Figure 7. The event log contains 1310 cases. Each event describes the progress of an individual patient. The timestamps are recorded for events. The log contains errors of missing events and also imprecisions in documentations (e.g., the timestamps are sometimes rounded to 5-minutes). Our assumption is that we can detect deviations from the control flow with conformance checking techniques [18,4,7], and therefore limit our

[3] see `StochasticNet` package in `http://www.promtools.org`

evaluation to the subset of 570 structurally fitting cases. The 570 cases contain 6399 events, which are assigned a timestamp each.

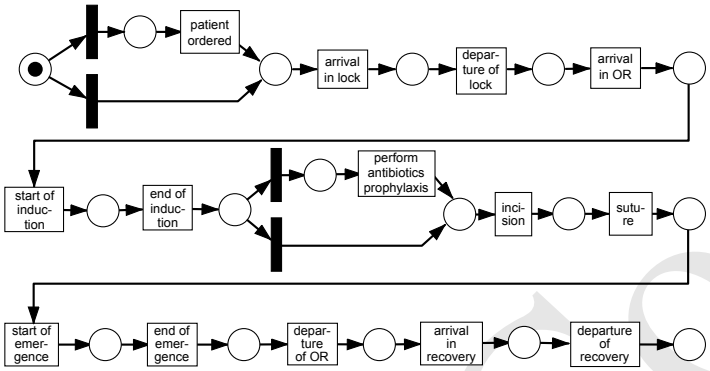


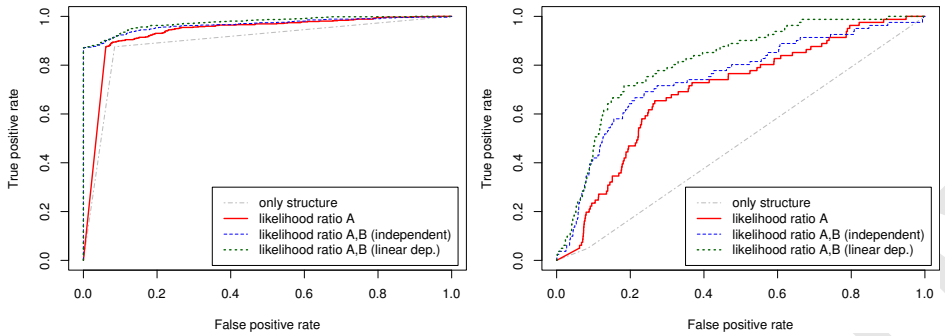
Fig. 7: Surgery model of a Dutch hospital. Most activities are in a sequential relationship.

We cannot be sure from the dataset alone, which outliers are due to measurement errors, and therefore perform a controlled experiment. We insert manual errors according to a Gaussian normal distribution with mean=0, sd=1/3 of the average process duration. We perform a 10-fold cross validation, to make sure that the duration distributions do not contain the original values. In the evaluation phase, we apply our approach to identify these errors. As described in the previous section, the approach should be able to identify errors based on the probability density of the original distribution. Furthermore, it should be able to identify many errors as obvious outliers, as a measurement error often causes a change in the ordering of events, i.e., leading to structural errors. Finally, some errors should not be detectable, because they may be very low or even 0. Other errors will be detected depending on the density region that they fall into. Here, chances are better, if their value becomes an outlier according to the error distribution.

Figure 8 shows the different receiver operating characteristic (ROC) curves for different prediction models:

- The solid line (in red) represents a model that predicts an error based on the likelihood that A is erroneous in two consecutive events A, B. This model ignores B and corresponds to a single-window approach.
- The dashed line (in blue) corresponds to model that predicts an error based on the likelihood that A is erroneous and B is erroneous, independently of each other.
- The dotted line (in green) stands for a model that predicts based on the likelihood that both A and B are erroneous, when linear dependency is assumed.
- The baseline model can predict only structural anomalies

The plot in Figure 8a, is computed from all events (with and without structural anomalies). In Figure 8b, events with structural anomalies are excluded. As it can be seen, the model that assumes a linear relationship between the errors of A and B (e.g., a positive error in one event is a negative error in the following event) performs best. This



(a) Results for all events.

(b) Results for events without structural anomalies.

Fig. 8: Receiver operating characteristic (also ROC curve) for identifying inserted errors.

model achieves an astounding area under the curve (AUC) of 97.5%, when applied to all events (i.e., with and without structural anomalies, see plot on the left).

The next predictor with satisfactory performance is the one that assumes independence between A and B given the possibility of an error. This model corresponds to a Naive Bayes prediction model. Despite its good performance its ROC curve is consistently below the more advanced predictor that takes dependencies into account.

The single duration window model that is based on the likelihood ratio of A being erroneous is already quite good, but it cannot distinguish between the error being caused locally or by a neighbor event.

Finally, the baseline predictor recognizes that two (or more) activities have been swapped in order, but it cannot determine which one is causing the error. This highlights once again the need for more advanced methods that solve this issue by using timing

	All events	Without structural anomalies
only structure	0.8953294	0.4823185
likelihood ratio A	0.9337065	0.6960063
likelihood ratio A,B (independent)	0.9671142	0.7490936
likelihood ratio A,B (linear dep.)	0.9753948	0.8122718

Table 1: Areas under the curve (AUC) for Figure 8. The AUC is a prediction quality measure that represents the ranking accuracy with respect to a specific scoring function. An ideal ranking (i.e., with AUC = 100%) would rank all positives on top of all negatives, thus enabling a clear separation between the two classes. As it can be seen, the score corresponding to the likelihood ratio of A and B, when they are assumed to be linearly dependent, yields the highest ranking accuracy. Moreover, when applied to all events (i.e., with and without structural anomalies, see plot on the left), the same method achieves an astounding AUC of 97.5%.

information and reasonable dependency assumptions. In this sense, the suggested method is a considerable improvement over state-of-the art techniques in conformance checking.

Note that the assumption of a normally distributed error turns out to be quite reasonable, because the variance is quite high, i.e., ~ 60 minutes (which on average corresponds to one third of the entire process duration). Furthermore, in the boundary of an activity, if the Gaussian representing its duration is flat enough to resemble a uniform distribution from the previous event to the successor event, the reasoning remains sound from a probabilistic perspective. Therefore, we do not expect major differences when using a uniform error distribution instead of a Gaussian.

We conducted further experiments with different kinds of distributions, of which we only present condensed insights due to space restrictions. The exponential distribution has only one tail, which makes detection of measurement errors more difficult than in the normally distributed case, where it is possible to detect positive as well as negative measurement errors. The general insight is that the stronger the signal-to-noise ratio becomes, the easier it is to detect measurement errors. For example, we can detect all measurement errors > 0, if the timed model is deterministic. It is almost impossible, however, to detect measurement errors of a uniformly distributed activity. Fortunately, real processes are seldom so extreme and when manual process activities are conducted, these activities tend to be rather normally or log-normally distributed.

6 Conclusion

In this work we focused on temporal aspects of anomalies in business processes. Preliminary evaluation on synthetic and real process data shows that the suggested method reliably detects temporal anomalies. Furthermore, it is capable of identifying single measurement errors by exploiting knowledge encoded in the process model. In the experimental evaluation, a large share of inserted errors were detected, even in real process data. The application of our approach to resources should be relatively straight-forward, but standard outlier detection techniques already yield good results [11]. The method is implemented in the open source framework ProM.

We expect the experimental findings to generalize to sensor-based measurements with corresponding changes in the assumed delay distributions. For example, in the case of sensors, an exponential distribution of delays might be more meaningful. However, an exact investigation of the impact of such distributions on the reliability of the suggested method is part of our future work. Other points on our future work agenda are the comparison of the method with other machine learning techniques and its extension to detect multiple erroneous events.

References

1. Wil M. P. van der Aalst. Verification of Workflow Nets. In *ICATPN'97*, volume 1248 of *LNCS*, pages 407–426. Springer, 1997.
2. Wil M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

3. Wil M. P. van der Aalst, Arya Adriansyah, Ana Karla Alves de Medeiros, et al. Process Mining Manifesto. In *BPM Workshops*, volume 99 of *LNBIP*, pages 169–194. Springer, 2012.
4. Arya Adriansyah, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Conformance Checking Using Cost-Based Fitness Analysis. In *EDOC 2011*, pages 55–64. IEEE, 2011.
5. Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41(3):1–58, July 2009.
6. Jonathan E. Cook, Cha He, and Changjun Ma. Measuring Behavioral Correspondence to a Timed Concurrent Model. In *ICSM'01*, pages 332–341. IEEE, 2001.
7. Fábio de Lima Bezerra and Jacques Wainer. Algorithms for Anomaly Detection of Traces in Logs of Process Aware Information Systems. *Inf. Syst.*, 38(1):33–44, 2013.
8. Guido Governatori, Zoran Milosevic, and Shazia Sadiq. Compliance Checking between Business Processes and Business Contracts. In *EDOC '06*, pages 221–232, 2006.
9. Frank E. Grubbs. Procedures for Detecting Outlying Observations in Samples. *Technometrics*, 11(1):1–21, 1969.
10. Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2nd edition, 2006.
11. Ming C. Hao, Daniel A. Keim, Umeshwar Dayal, and Jörn Schneidewind. Business Process Impact Visualization and Anomaly Detection. *Information Visualization*, 5(1):15–27, 2006.
12. Niels Lohmann, H. M. W. (Eric) Verbeek, and Remco Dijkman. Petri Net Transformations for Business Processes - A Survey. In *Transactions on Petri Nets and Other Models of Concurrency II*, volume 5460 of *LNCS*, pages 46–63. Springer Berlin Heidelberg, 2009.
13. Emanuel Parzen. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.
14. Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, USA, 2000.
15. Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Technische Hochschule Darmstadt, 1962.
16. Andreas Rogge-Solti, Wil M. P. van der Aalst, and Mathias Weske. Discovering Stochastic Petri Nets with Arbitrary Delay Distributions From Event Logs. In *BPM Workshops*, volume 171 of *LNBIP*, pages 15–27. Springer, 2014.
17. Andreas Rogge-Solti, Ronny S. Mans, Wil M. P. van der Aalst, and Mathias Weske. Improving Documentation by Repairing Event Logs. In *The Practice of Enterprise Modeling*, volume 165 of *LNBIP*, pages 129–144. Springer Berlin Heidelberg, 2013.
18. Anne Rozinat and Wil M. P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Inf. Syst.*, 33(1):64–95, 2008.
19. Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1996.
20. Edward H. Simpson. The Interpretation of Interaction in Contingency Tables. *Journal of the Royal Statistical Society*, Series B:238–241, 1951.
21. Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, second edition, 2012.
22. Andreas Wombacher and Maria-Eugenia Iacob. Estimating the Processing Time of Process Instances in Semi-structured Processes–A Case Study. In *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, pages 368–375. IEEE, 2012.
23. Dit-Yan Yeung and C. Chow. Parzen-Window Network Intrusion Detectors. In *ICPR'02*, volume 4, pages 385–388. IEEE, 2002.