A Dynamic Threshold Algorithm for Anomaly Detection in Logs of Process Aware Systems

Fábio Bezerra[1], Jacques Wainer[2]

[1] Universidade Federal Rural da Amazônia
fabio.bezerra@ufra.edu.br
[2] Universidade Estadual de Campinas
wainer@ic.unicamp.br

Abstract. In the last years, companies have adhered to PAIS (Process Aware Information Systems) for supporting the control of their businesses. However, while normative PAIS may compromise the competitiveness of these companies, flexible PAIS are a risk for security. In order to re-balance that trade-off, we present a new approach for anomaly detection in logs of PAIS. It is an algorithm based on conformance threshold that is dynamically defined. The algorithm was evaluated on two datasets of artificial logs (one with 360 complex logs, and other with 1800 simpler logs), with different profiles on the number of anomalous traces and the number of times each anomalous traces was present in the log. We also carried out a comparative study with a naive approach for anomaly detection that marks as potential anomalies traces that are infrequent in the log.

Categories and Subject Descriptors: H.2 [**Database Management**]: Miscellaneous; H.3 [**Information Systems Applications**]: Office Automation; K.6 [**Management of Computing and Information Systems**]: System Management

Keywords: anomaly detection, business analysis, process aware systems, process mining

1. INTRODUCTION AND MOTIVATION

Management trends in the early 1990's largely motivated the adoption of **P**rocess **A**ware **I**nformation **S**ystems (PAIS), or simply Process Aware Systems (PAS), by organizations[Dumas et al. 2005]. This scenario represents a shift from data to process-oriented systems, which clearly separates business process logic from application programs, facilitating redesign and extension of process model. Moreover, such a separation supports the coordination and control of business, for example, either guarantying that the execution of activities obey its prescribed control-flow definition, specially in normative systems like WFMS, or providing tools that identify bottle-necks paths. On the other hand, the business process control of competitive companies should not be supported by normative tools like a classical production WMS (Workflow Management System). These companies demand a flexible automation of their business processes, for they need to respond rapidly to new market strategies or new business models. However, a flexible system may be vulnerable to fraudulent or undesirable executions. Therefore, there is clearly a trade off between flexibility and security.

In this context, process mining appears a promising technology for process analysis and discovery. The idea of process mining is to discover, monitor and improve processes by extracting knowledge from event logs readily available in today's systems[Aalst et al. 2012]. Process mining is a technology that may consider four different scopes or perspectives: control-flow, time, data, and organizational. Moreover, there are three types of process mining: discovery (from an event log it outputs a model), conformance (analysis how much reality, the log, conforms to the model), and enhancement (improve

a model based on the reality, the log).

This research is restricted to *control flow* perspective, but it considers both discovery and conformance types of process mining. It presents results in detecting anomalies in logs of PAS, where the anomaly is detected solely based on the sequence and choices of activities that took place in that anomalous execution. For example, a trace may be an anomaly because a pair of its activities are not in an appropriate sequence. Of course, the anomalous nature of a case may be derived from the values involved in some of the activities, or because of the people who executed some of the activities, or because of time to perform an activity or the whole process was greater or less than normal. We call these examples as *data*, *organizational*, and *time* anomalies respectively, to match the other aspects of process mining[van der Aalst 2011; Aalst et al. 2012].

Previous research on process mining, in the context of anomaly detection, proposed three algorithms based both on conformance analysis and process discovery[Bezerra and Wainer 2012; 2011; de Lima Bezerra 2011]. For two of these algorithms, *Iterative Algorithm* and *Threshold Algorithm*, a trace is an anomaly if its degree of conformance is *lower* than a given threshold. While in the third algorithm, *Sampling Algorithm*, which presented the best results [Bezerra and Wainer 2012], a trace is an anomaly if it does not conform 100% with a process model discovered from a sampling of the log. These results are based on artificial logs, since there is no publicly available datasets or benchmarks for anomaly detection in processes.

This work presents results of a new threshold approach for anomaly detection in PAS. This a new approach is an extension of the previous *Threshold Algorithm* version[Bezerra and Wainer 2012], but as opposed to the *Threshold Algorithm*, it does not require the threshold to be an externally determined parameter of the algorithm. This new algorithm dynamically evaluates the threshold based on the distribution of sample means, which is inspired by central limit theorem[1]. This new algorithm represents our effort to develop a solution with a higher efficacy than the results reported for *Sampling Algorithm* either in [Bezerra and Wainer 2012] and in [de Lima Bezerra 2011]. In addition, similarly to previous anomaly detection algorithms, it is not a fully automatic detection method. That is, anomalous traces, once discovered, must be analyzed to find out if indeed they are examples of incorrect executions or if they are acceptable but uncommon executions. And if they are found to be incorrect executions, the reasons for and consequences of these executions must be further investigated. Therefore, it is an *a posteriori* method, a first automated step towards a more comprehensive security auditing practice for flexible PAS.

Some important requirements were considered in the development of algorithms: on one hand, (i) anomalous traces that are an indication of fraud may have serious consequences to the business, so the algorithm must have a very low false negative rate; on the other hand, (ii) flagged anomalous traces are forwarded to an analyst, which is a costly process, so the algorithm must have low false positive rate; finally, (iii) a low false positive rate is less important than a low false negative rate.

This work is organized in the following way. In Section 2 we present the definition of anomalous trace supported by the algorithm, which is presented in Section 4. In Section 3 we present previous research on process mining and anomaly detection on PAS. In Section 5 we present the assessment carried out using two synthetic datasets, while is Section 6 we present the results of assessment. Finally, in Section 7 we present some conclusion and directions for future work.

2. TOWARDS A FORMAL DEFINITION OF ANOMALOUS TRACE

There are many semantics associated with the definition of anomaly. Only to cite some examples, an anomaly can be an **exceptional** execution, a **noise** in the log, possibly caused by system failure or

[1]The central limit theorem states that when sampling from a large population of any distribution shape, the sample means have a normal distribution whenever the sample size is 30 or more[Larson and Farber 2003].

error in data input, or even a **fraud** attempt. An exception characterizes an abnormal or unusual execution, but it can be supported by the business. Whereas a fraud attempt and an operational error are unusual executions that provoke undesirable results to business. However, despite the different meanings associated with the term anomaly, there are some common sense definition as follows: (i) an anomaly in a rare or infrequent event; or (ii) it is a deviation from a normal form or rule; or (iii) it is an unexpected result; or (iv) it is an state outside the usual range of variation from the norm.

Nevertheless, a precise definition of normal, norm, or rule is difficult, if not impossible, in some application domains of PAS (e.g. health care systems). Besides, it is a naive approach to classify a trace considering only its frequency in the log because some paths in a process model can be enacted more frequently than others, so it is probable that some "normal" traces be infrequent. For that reason, we believe that anomaly detection method should be based not only on the frequency of traces.

In this research we assume that anomalous traces are traces that are not instances of a process model that "best explain" the non-anomalous traces in the set. For instance, given a set of traces T, one can clearly say that there are no outliers since both the most general and the most specific models explain all traces in T, but are these specific and general models the models that "best explain" the traces? We assume that there are ways for partitioning the set T into two groups, anomalous and non-anomalous (or normals), so that the model mined from the normal set is "reasonable" (not too complex, not too generic, not too specific) and it does not fit the traces in the anomalous set. Thus, we present below some formal definitions that lead us towards this assumption.

Throughout this article the term **trace** will be used to refer to an execution path (or process instance) of a business process model, and it represents the order that the activities of this path were completed. Thus, a trace $[a\ b\ c\ d\ e]$ indicates that activity a finished before activity b, and that activity b finished before activity c, and so on (Definition 1). Besides, Definition 2 is a formal definition for a set of traces which we call **log**.

DEFINITION 1. *Trace.*
Given that A is a set of activities. Then, a trace t represents a sequence of activities such that $t \in A^*$. That is, assuming that A is an alphabet, and A^* denotes all possible words over A, then t is a word based on this alphabet.

DEFINITION 2. *Log.*
Given $T = \{t \mid t \in A^*\}$ a set of all traces defined over A and $T' \subseteq T$, then a log L is a multiset defined over T', where $L = \{(t,n) | t \in T' \ \wedge n \in \mathbb{N}\}$.

Normally the *log* that will be applied for anomaly detection needs to be filtered, for removal of those traces that are clearly anomalous instances or removal of those activities that are not important for the analysis. For example, when a log is generated for analysis, there may be a lot of instances in execution that were not concluded yet; these instances can not be provided for the anomaly detection algorithm. Definition 3 present a formal definition for the filtered log. While in Definition 4 we formally present what we call conformance degree between a model and a log, that is, how much of log adheres to the model.

DEFINITION 3. *Filtered Log.*
Given:

—*a log L (Definition 2);*
—*a set A^S of activities that were filtered from the log (scoped), such that $A^S \subseteq A$;*
—*a function* filter(t, A^S) *that removes all activities in a trace t that are not in A^S;*
—*a boolean function complete(t) that outputs* false *if t is an incomplete trace and true if t is complete, given that initial and final activities are known.*

Then, a filtered log $L^S \subseteq L$ is a multiset of traces t based on activities of A^S, as follows:

$$L^S = \{filter(t, A^S) \mid (t, n) \in L \wedge complete(t)\}$$

DEFINITION 4. *Conformance Degree of Log.*
It is a function $f : \{(M, L)|M$ is a model $\wedge$ L is a log$\} \rightarrow [0, 1]$ that measures the degree of conformance *between the model M and the log L. This function outputs a continuous value, ranging from 0 to 1. The value 1 means that the log L completely conforms to the model M, while the value 0 means that the log L does not conform to the model M.*

Then, in order to select the anomalous traces from the log we can initially suppose that every trace from the log may be an anomalous trace. However, some heuristics should be applied in order to select those instances that are more likely to be an anomaly. For instance, those frequent traces cannot be an anomaly, by virtue of our own definition of anomaly as an infrequent case. For that reason we present below, in Definition 5, what we call candidate anomalous trace. A trace is a candidate anomalous trace if its class has a frequency less or equal a given frequency. For the purpose of this work, we adopt a frequency of 2% to classify a trace as a candidate anomalous trace. The choice of 2% as the threshold for anomaly is arbitrary but it reflects the assumption made in this research that anomalies are fraud (and thus the requirement of very low false negative rates). The 2% threshold seems reasonable for this fraud perspective - a business in which more than 2% of the processes are fraudulent are indeed in trouble.

DEFINITION 5. *Candidate Anomalous Trace.*
Given:

—a filtered log L^S (Definition 3);
—a set $C_L = \{c|(c, n) \in L^S\}$ of classes of trace in the log L^S;
—a real value $x \in [0, 1]$;
—$s_L = \sum_{(c,n)\in L} n$ the number of traces from the log *L^S;*
—$f_c = \frac{n}{S_L}$ the frequency of the class c in the log L^S, where $(c, n) \in L^S$.

Then, t^c is an anomalous candidate trace if it is in the set T_C, as follows:

$$T_C = \{c \in C_L \mid f_c \leq x\}$$

Once we have the set of candidate anomalous traces, its elements will be classified as anomalous if they satisfy the Definition 6. It indicates that try to fit an anomalous trace in a "normal" process model would require a lot of modifications in the model. Then, the conformance of the log with the anomalous trace is much smaller than the conformance of the log without the anomalous trace.

DEFINITION 6. *Anomalous Trace.*
Considering the following items:

—a filtered log *L^S (Definition 3);*
—a set T_C of candidate anomalous traces (Definition 5);
—a set $T_A \subseteq T_C$ of anomalous traces;
—a set $T_N \subseteq L^S$ of normal traces;
—L^S can be partitioned into two multisets A (anomalous) and N (normal) such that $A \cup N = L^S$ and $A \cap N = \emptyset$;
—a function $f(M, L)$ (Definition 4);
—a model M_N mined from the log T_N, where $f(M_N, T_N)$ is the maximum;

—*a model* M_L *mined from the log* L^S*, where* $f(M_L, L^S)$ *is the maximum;*

—*a partial order "make more sense" between models denoted as* $M_1 \succ M_2$ *which indicates that* M_1 *"make more sense" than* M_2

Then t^a *is an anomalous trace if it is in the set* T_A*, defined as follows:*

$$T_A = \{t^c \in T_C \,|M_N \succ M_L\,\}$$

3. RELATED WORK

Data mining community has published a large and growing body of literature in anomaly detection. To cite a few as an example, in [Donoho 2004] the author presents how data mining techniques can be used to early detect inside information in option trading. In [Fawcett and Provost 1997] the authors present a system which is used to detect fraudulent usage of a cellular phone (cellular cloning). Moreover, disease outbreak detection has been proposed by detecting anomalies in the event logs of emergency visits [Agarwal 2005], or the retail data for pharmacies [Sabhnani et al. 2005]. There are solutions concerned with the intrusion detection in networks [Lee and Xiang 2001; Noble and Cook 2003]. Other efforts are concerned with the detection of fraudsters in auctions or e-commerce sites [Pandit et al. 2007]. In the next sections we reported how process mining have been supported the anomaly detection field in the context of PAS.

3.1 Process Discovery

Process discovery is a method used to reconstruct a process model from a log generated by a system. Such a technology is an alternative to construct models from scratch, and in the last fourteen years it has raised the attention of researchers around the world [van der Aalst et al. 2003; de Medeiros et al. 2003; van der Aalst et al. 2004]. For this work, process discovery algorithms are important to help us find a model that will be used as a classifier of traces as anomalous or normal.

The field of process discovery was first coined in the context of software processes. Cook and Wolf, in [Cook and Wolf 1998], present process discovery as a tool to support the design of software processes because it is a hard, expensive, and a error prone activity, specially for big and complex processes. Also a forerunner work in process discovery, the paper of Agrawal et al, in [Agrawal et al. 1998], present an algorithm that mine models with three properties: completeness, minimality, and no redundancy. Complete in the sense that a process model may contain all task and dependencies between them that there is in the log; minimal in the sense that the mined process has the minimum set of structural elements; and no redundancy in the sense that the process model does not play an instance by different ways.

A lot of process discovery approaches have been proposed in the last years [van der Aalst et al. 2003; van der Aalst et al. 2004; van der Aalst and Weijters 2004; Schimm 2004; Herbst and Karagiannis 2004; Wen et al. 2006; Hammori et al. 2006; de Medeiros et al. 2006; Wainer et al. 2005]. Among the recent process discovery approaches, the most broadcast one is the $\alpha-$algorithm [van der Aalst et al. 2003; van der Aalst et al. 2004; van der Aalst and Weijters 2004]. The efficacy of that algorithm was formally proved for a class of process models, the WF-Nets (*Workflow Net*), which are petri nets that require: (i) a single Start place, (ii) a single End place, and (iii) every node must be on some path from Start to End. However, such an algorithm has some limitations, for example, to mine short loops in the model and non-free-choice constructs, and for that reason some extensions were developed. For example, in [Wen et al. 2006], the authors present an extension version of $\alpha-$algorithm for solving the detection of implicit dependencies between tasks from event logs.

Hammori et al. present an interactive process discovery algorithm in [Hammori et al. 2006]. The resulting model is represented in block-structured manner. This work report an implementation of

the approach in the ProTo system, which is an alternative of InWoLve [Herbst and Karagiannis 2004] for supporting interactive discovery. Actually, ProTo uses InWoLve as its kernel.

Schimm presents a discovery approach that discovers a model after merging or rewriting the traces of log with a set of formal rules (axioms) of an workflow algebra [Schimm 2004]. Besides, different from more disseminated approaches, the output model is a block-oriented representation. It defines that each workflow model consists of an arbitrary number of nested building blocks. Also, the models meet three requirements: completeness, specificity (it does not introduce additional tasks or spurious dependencies between tasks), and minimality. Similar to Schim's approach, Wainer et al., in [Wainer et al. 2005], argue that the process discovery problem is not well defined because a huge number of solutions totally complied with a log can be inferred, then they present a sketch of a process discovery algorithm that characterizes the stated problem. That algorithm is an incremental discovery approach, where a model is created trace by trace until all traces from the log are used. For that reason, the authors suggest a reformulation of the problem that consider the selection of the best or more appropriate model for a given log.

Some process discovery tools may deal with noise in the log [Agrawal et al. 1998; van der Aalst et al. 2003; Cook et al. 2004; Pinter and Golani 2004; Herbst and Karagiannis 2004; Hammori et al. 2006]. These approaches are limited to the frequency evaluation of dependency relation between two activities. For example, infrequent dependency relations between two activities may not be modeled in the resultant process model. Nevertheless, a more sophisticated and promising approach was proposed in [de Medeiros et al. 2006; de Medeiros 2006], it is called *genetic mining*. It is based on genetic algorithms, which is a search for a solution (an individual) that satisfies a selection criteria (fitness function), and the individuals are defined based on some genetic operators (eg.: crossover, mutation, and elitism).

The use of an appropriate process discovery tool is crucial for the accuracy of anomaly detection. Bad results in the accuracy of an algorithm for anomaly detection can be associated to bad discovered models, since the models are actually used to support anomaly detection. For example, if the miner outputs a too generic model (the flower model), it can classify as normal the anomalous traces, while if it outputs a too specific model (a model that is a branch for each class of trace), it can classify as anomalous the normal trace not observed in the log the generated the model.

For the purpose of this work we utilized four different process discovery tools that can be used as input for the anomaly detection algorithm (the miner parameter): (i) *alpha* algorithm[van der Aalst et al. 2003; van der Aalst et al. 2004]; (ii) *alpha++* that is an extension version of alpha algorithm with support for non-free-choice constructs[Wen et al. 2006]; (iii) *heuristic miner* algorithm that is also an extension of alpha algorithm, but it supports noise in the log and it considers the frequency of dependencies between activities to represent them in the log[Weijters and van der Aalst 2003; Weijters et al. 2006], so it is a more robust approach for process discovery; and finally (iv) *multiphase algorithm* that outcomes an EPC model[van Dongen and van der Aalst 2004].

3.2 Anomaly Detection

All process mining methods mentioned in previous section are mainly concerned with the modeling of normal behavior, yet some of them also deal with noisy logs. Therefore, abnormal behavior was not deeply study by process mining community, although it is a clearly important subject to the development of more accurate *audit systems* or systems that *handle exception* cases. For example, it would be important to understand how "special clients" are assisted in exceptional situations, so some process improvements would be suggested to handle such occurrences in a more appropriate way. Moreover, in the case of audit systems, it would be a clue for identification of fraudsters. Therefore, in order to fill this gap, more recent researches have been addressing the problem of identifying anomalous trace in logs of PAS [van der Aalst and de Medeiros 2005; Yang and Hwang 2006; Bezerra and Wainer 2008a; 2008b; Bezerra et al. 2009; Bezerra and Wainer 2011; 2012].

In [van der Aalst and de Medeiros 2005], Aalst and Medeiros present two anomaly detection methods that are supported by α-algorithm. A drawback of this work is that it demands a known "normal" log, which is used to mine a process model that will classify the traces of an audit log. Unfitness instances are the anomalous traces. However, a known "normal" log may not be available in applications domains that demands a high flexible support because the normal instances change constantly. In [Yang and Hwang 2006] the authors present a framework for automatic construction of a model for detecting health care fraud and abuse. In that work clinical pathways are used to construct a detection model, whose features are based on frequent control-flow patterns inferred from two datasets, one with fraudulent instances and other with normal instances. Similarly, a dataset of normal and fraudulent instances may not be available in some applications, which is clearly a serious limitation.

Closer related researchers to this work have been published in [Bezerra and Wainer 2008a; 2008b; Bezerra et al. 2009; Bezerra and Wainer 2011; 2012]. In [Bezerra and Wainer 2008a; 2008b], we present some approaches based on incremental mining[Wainer et al. 2005], but these algorithms can not deal with longer traces and/or logs with various classes of traces. Then, in order to deal with such constraints, we begun to develop other solutions based on process mining algorithms available in *ProM framework* [Bezerra et al. 2009; Bezerra and Wainer 2011]. In [Bezerra et al. 2009], we proposed an anomaly detection model based on the discovery of an "appropriate process model", but we believe that it has two drawbacks: (i) it is not an automated solution, since there is not an implementation for it yet; and (ii) it considers a search for an appropriate model, which is still an open definition in the process mining field. In [Bezerra and Wainer 2011], we started to use the process discovery and conformance algorithms from *ProM framework* for implementing the anomaly detection algorithms.

More recent research findings into anomaly detection on PAS have been reported in [de Lima Bezerra 2011; Bezerra and Wainer 2012]. Among the algorithms reported in these work, the *Sampling Algorithm* presented the best results. Such an algorithm is based on the following materialization of the basic intuitions: since anomalies are infrequent, a random sample of the trace-instances from the log will likely not include an anomaly. Then, models mined from this sample will probably not include the anomalies as instances,and traces that are not instances of this model can be considered anomalies.

The new anomaly detection algorithm proposed in this work is an attempt to implement a better solution for the anomaly detection problem on PAS. Our intention was to develop an algorithm more efficient than the *Sampling Algorithm*. The approach for anomaly detection proposed here represents an extension of the *Threshold Algorithm* also reported in [de Lima Bezerra 2011; Bezerra and Wainer 2011; 2012], which uses process mining tools for process discovery and process analysis for supporting the detection.

3.3 Process Analysis

In [Rozinat and van der Aalst 2005; 2008], Rozinat and Aalst present different dimensions for evaluation of process models. Beyond fitness dimension, which describe how much of log can be correctly played in a model, the authors also present the precision and structure dimensions. Precision dimension indicates how much behavior is supported by a model, whereas structure dimension indicates how much complex is a model. The combination of these metric help us assess mined process models, giving us confidence of among mined models which one "better" describes a log.

Genetic mining uses some similar metrics to evaluate the individuals that will be select to comprise and be the matrix of next generation of individuals [de Medeiros 2006]. In [Rozinat et al. 2007] and [van Aalst et al. 2010], the authors present a deeper analysis of different assessment metrics, comparing different evaluation aspects of each metric (e.g. input, output range, dimension of assessment, and computational complexity).

For the purpose of this work we utilized four different process analysis tools that can be used as

input for the anomaly detection algorithm (the evaluator parameter): (i) *fitness*, which assess the portion size of the log that can be correctly played by the model, that is, the degree of completeness; (ii) *behavioral appropriateness*, which measures the degree of predictability to support the execution of unseen trace in the log; (iii) *structural appropriateness*, which assess the degree of complexity; and (iv) *appropriateness*[Bezerra et al. 2009], which was suggested to be a balance between structural and behavioral appropriateness. The *appropriateness* metric is not directly available in the ProM framework, but it is evaluated based on methods from conformance check plugin of ProM framework.

4. DYNAMIC THRESHOLD ALGORITHM

In this section we present the algorithm that represents a concrete implementation of previous definitions. First, in Section 4.1 we highlight the main points of the original idea. Then, in Section 4.2, we present the new algorithm, which is supported by some process mining tools reported in Section 4.3.

4.1 Threshold Algorithm: Original Idea

In the threshold approach, a candidate trace will be an anomaly if the conformance between the log and the model mined from the same log, but without instances of the trace under analysis, is lower than a given threshold provided as input for the algorithm[Bezerra and Wainer 2011; de Lima Bezerra 2011]. For each candidate anomalous trace, the algorithm is executed as follows: (i) it gets a new log removing occurrences of the candidate trace from the original log; (ii) it creates a process model for this new log; (iii) evaluates the conformance between the model and the original log, that is, the log with occurrences of the candidate anomalous trace; and (iv) test if the conformance is lower than a given threshold. Anomalous traces are those traces that has a conformance value below a threshold provided as input for algorithm.

As reported in Section 3, the conformance check metrics available on ProM assess different characteristics of a model, for example, since the degree of complexity of a model until the degree of completeness of a model for a log. Therefore, the use of an appropriate metric is important for the accuracy of detection, as well as the process discovery tool.

Other parameter that may impact the accuracy of detection is the threshold factor. Lower values indicate lower tolerance for the degree of conformance – there may have normal traces classified as anomalous (False Positive). On the other hand, higher values indicate higher tolerance – there may have anomalous traces classified as normal (False Negative). Therefore, to choose a good value to the threshold is not easy. Following, we propose an extension to the idea of threshold algorithm that does not consider the threshold value as input, but **it dynamically evaluates it**.

4.2 Algorithm: Extended Idea

We present in Algorithm 1 the anomaly detection method that we called **Dynamic Threshold Algorithm**, for it is inspired in the threshold algorithm, described in Section 4.1, but considering a threshold value dynamically calculated (Line 18 in the Algorithm 1). The number of parameters is the main difference of this algorithm when compared with its previous version, for it does not use as input a threshold conformance value for classifying a trace. Now the algorithm has three input parameters: the log, the process discovery miner, and the conformance analysis evaluator.

First of all, as defined in Definition 5, the algorithm selects the candidate anomalous traces (Line 4). Then, a process model is discovered trough the process discovered miner and the log provided as input for the algorithm (Line 7). Such a model represents a "description" of all the traces of the log, and it is used as a reference for evaluating the conformance values. Next, the *Dynamic Threshold Algorithm* evaluates the threshold value that will be used for classification of the candidate traces. Such an evaluation is based on the central limit theorem, which defines that the distribution of an average

Algorithm 1: Dynamic Threshold Algorithm

Input: A log L, which is a set of traces generated by a PAS.
Input: A process discovery algorithm PD.
Input: A conformance assessment algorithm CA.
Output: A set of traces T^A that was classified as anomalous traces.

```
1  Define a set T with all classes of traces from the log L;
2  Define a set T^C = {} used to contain the anomalous candidate traces;
3  Define a set Conformances = {} used to contain the evaluated conformances;
4  foreach t ∈ T do
5      if Frequency of t into the log ≤ 2% then
6          Include t into set T^C;
7  model ← Miner(L, PD);
   // 50 samplings
8  numberOfSamplings ← 50;
   // each sampling has at least 30 traces
9  if Size of L * 2% >= 30 then
10     numberOfTraces ← Size of L * 2% >= 30;
11 else
12     numberOfTraces ← 30;
13 while numberOfSamplings > 0 do
14     sampling ← Sampler(L, numberOfTraces);
15     tempValue ← Conformance(model,sampling,CA);
16     Include tempValue into set Conformances;
17     numberOfSamplings ← numberOfSamplings − 1;
18 threshold ← Mean(Conformances) −1* Std(Conformances);
19 foreach t ∈ T^C do
20     tempValue ← Conformance(model,t,CA);
21     if tempValue < threshold then
22         Include t into set T^A;
```

tends to be normal, even when the distribution from which the average is computed is decidedly non-normal[Larson and Farber 2003]. To match the definition of such a theorem, which supports the calculation of the threshold value, the algorithm plays as follows:

—In Line 8 we define how many samplings are going to be used for evaluation of the threshold value. Increasing this number improves the shape of distribution to normal curve, but it demands a higher processing cost, for it will be more samplings for the evaluation of conformance. Because we assume the frequency of candidate traces of 2%, we arbitrarily use 50 (1/0.02) samplings of the log to define the sampling distribution of means.

—In Line 9 we define the number of traces of the sampling, that is, how many traces are randomly selected from the log to comprise a sampling. For populations with normal distribution, whatever value could be defined. However, because we do not know a priori what is the conformance distribution of traces in the log, we guarantees a minimum of 30 traces. Otherwise, this value is 2% of size of the log (again we arbitrarily adopted the frequency of candidate traces to support the choice).

—In Line 13 we define a set of conformance values, one for each sampling of the log. A conformance value represents a mean of conformances of every trace in the sampling. From these values the

threshold is evaluated.

—In Line 18 we calculate the threshold value, which is based on the mean and standard deviation of conformance set.

Once we have the threshold value, for each candidate trace (Line 19), we classify as anomalous those ones that has a conformance value (Line 20) below the calculated threshold (Line 21). Finally, we synthesize the differences between the algorithms in Table I.

Table I: Differences Between Algorithms

Threshold Algorithm	**Dynamic Threshold Algorithm**
It demands from the user a threshold value	The threshold value is not provide by the user, but it is dynamically evaluated
For each candidate trace a process model for classifying the trace is discovered	A unique process model for classifying the trace is discovered, and it is based on the whole log

4.3 Process Mining Support

The implementation of algorithm is supported by ProM Framework[2], which is an open source process mining tool, comprised of a set of process discovery and process analysis tools. It integrates the functionality of several existing process mining tools and provides many additional process mining plug-ins[van Dongen et al. 2005].

The Dynamic Threshold Algorithm is mainly supported by ProM tools in two points: (i) *first*, during the discovery of the process model; (ii) *second*, during evaluation of conformance between the model and the log. Such an assessment is based on a conformance check tool of ProM or a combination of them (for example, appropriateness is a balance between behavioral and structural metrics).

The tool used for process discovering is defined as an input parameter of the *Dynamic Threshold Algorithm*. In addition it may be one of the following: alpha, alpha++, heuristic miner and multiphase algorithm. Regarding the conformance evaluation tool, it also needs to be defined by an input parameter, and it may be one of the following: fitness, behavioral appropriateness, structural appropriateness and appropriateness.

5. ASSESSMENT PROCEDURES

We have assessed the the algorithm with two datasets, both referred in [Bezerra and Wainer 2012]. The *first dataset* has 360 logs with different configurations, supporting OR-blocks (exclusive), AND-blocks, LOOP-blocks and sequences – the procedure used to create this dataset is available as a appendix in [Bezerra and Wainer 2012]. The *second dataset* has 1800 synthetic logs, but they are simpler than the ones in the first dataset, for they contain less activities (at most 20 different activities in them), and they did not contain loops – the procedure used to create this dataset is available in [de Lima Bezerra 2011].

Two reasons influenced our choice for using synthetic data for the assessment. *First*, it is hard (perhaps inexact) to identify an anomalous trace in a real log, so it would impose some limitations on the assessment. For example, in [Pandit et al. 2007] the authors report some problems regarding the assessment of their anomaly detection system with real data. The problems are mainly related to subjective analysis of traces as an anomalous instance. *Last but not least*, a real dataset of process instances is not available.

[2]http://www.processmining.org

The assessment carried out in this work can be summarized in three steps, as follows: (i) first we played the algorithm with a short subset of logs, in other to get a tunning of parameters among 16 possibilities available for algorithms (four miner algorithms and four evaluator algorithms); (ii) since we have had the best set of parameters, we carried out the algorithm over the remaining logs (Section 6); (iii) finally we assessed and compared the results.

5.1 Comparison Strategies

An algorithm that classifies all the candidate traces as anomalous would get a recall of 1. On the other hand, its precision would be too low, mainly when there are few anomalous traces and many normal traces among the candidate set. Because it is possible to occur infrequent normal traces in the log (e.g. some paths of model are preferable than others, or application domains like health care and hospital whose each case may be different of others), such a procedure for anomaly detection is clearly very naive.

Therefore, it would be interesting to know if the *Dynamic Threshold Algorithm* is better than a simple, fast, but *naive approach*. In order to get the performance data from this naive approach we did as follows: (i) the *recall* is the constant value 1, since all anomalous occurrences in the set of candidate trace will be correctly classified as anomalous; (ii) the *accuracy* is the ratio between the number of anomalous and the number of candidate traces, since only the anomalous traces will be correctly classified; (iii) the *precision* is equal to accuracy because all candidate traces are classified as anomalous; and (iv) the *f-measure* is got from recall and precision values. Therefore, because we have such performance data we can compare all approaches.

5.2 Quality Metrics

In order to discuss the quality metric, let us briefly review the standard metrics for a binary classifier. The classes of a binary classifier are usually referred to as "positive" and "negative". In our case, "positive" refers to being an anomaly, and "negative" to being a normal trace. Furthermore:

—a true positive (TP) is an example (a trace in our context) that is classified as positive by the system and it is indeed positive.
—a false positive (FP) is an example that is classified as positive by the system, but it is really a negative example
—a true negative (TN) is an example that is correctly classified as negative by the system
—a false negative (FN) is an example that is classified as negative by the system, but it is really a positive example

The standard metrics for quality of a binary classifier are:

—accuracy = $(TP + TN)/(TP + TN + FP + FN)$
—precision = $TP/(TP + FP)$
—recall = $TP/(TP + FN)$

In the case of this work, because of the emphasis on anomalies as frauds, we want to the false negatives to be as small as possible, which implies that recall will be as close to 1 as possible. But using only recall as the metric of quality of the algorithms is inappropriate, since by definition, the naive algorithm will have recall equal to one. In this case, the naive algorithm will have the highest recall, but in turn it will result on the lowest precision - since all infrequent traces will be marked as positive. The f-measure family of metrics tries to balance (with different emphasis) both precision and recall [van Rijsbergen 1979]. They are defined as:

—f-measure = f1-measure = 2.precision.recall/(precision + recall)
—fβ-measure = $(1 + \beta^2)$.precision.recall/(β^2precision + recall)

F1-measure is the harmonic mean between precision and recall. The fβ-measure weights the recall by β when performing the harmonic mean. For $\beta > 1$, recall will be increasingly more important when performing the harmonic mean, the larger t β.

As we discussed above, under the fraud perspective, recall should be as high as possible, but precision should also be high. Thus we need a metric that places more emphasis on recall, and we arbitrarily opted for the f4-measure. But we will also report the f1-measure and accuracy. The accuracy calculations involve the true negative (TN) which will always be very high, so the figures for accuracy will be deceptively high. So we only report the accuracy for the candidate traces, that is, the traces with frequency lower than 2%. We also consider the f1- and f4-measures to be 0 when either precision or recall are 0.

5.3 Parameterization of the Algorithm

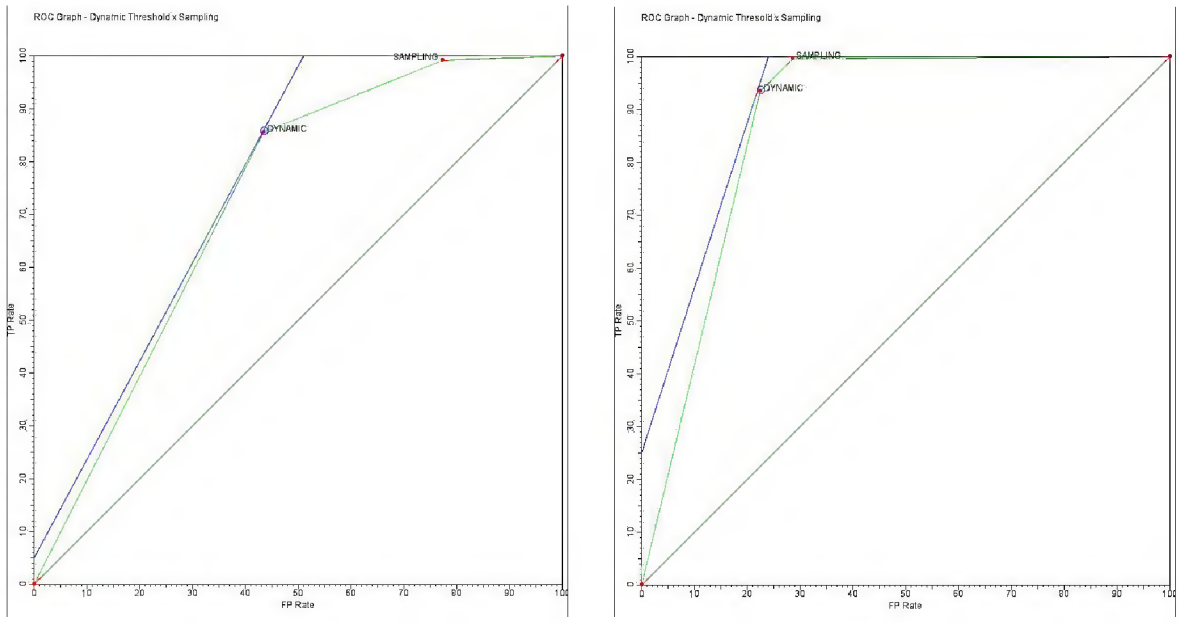
Considering the set of parameters that can be applied over the new anomaly detection algorithm, we can say that there are 16 detectors (detection solution), one for each combination of parameters (four process discovery and four conformance analysis algorithms). In order to get the best set of parameters, the logs were organized in two sets: (i) the **training group**, which were used to play the algorithm using all the combination of input values for identifying the best set of parameters; and (ii) the **test group**, which were the non observed set of logs used to play the algorithm, appropriately tuned by parameters.

Finally, once we played the algorithm over the *training group* of logs, we organized the results of parameterization in a descending way by f-measure, recall, and accuracy respectively in this order. The objective was to discovery a set of parameters that produced the best results, that is, higher values are better than smaller ones. Then, we defined the following parameters for the algorithm: **(i)** Process discovery algorithm: **Heuristic miner**[Weijters and van der Aalst 2003; Weijters et al. 2006]; **(ii)** Process model evaluation algorithm: **Behavioral appropriateness**[Rozinat et al. 2007; van Aalst et al. 2010]. Such a combination of parameters is related to a detector that had a performance over the training group, as follows: accuracy of 73,5%, f1-measure of 0.7206, and f4-measure of 0.8397. In Section 6 we present the results of carrying out such a detector over the testing group of logs.

6. RESULTS

The results reported in the following subsections are related to the *test group* of logs. Result data related to Naive, Threshold and Sampling algorithms were obtained from [Bezerra and Wainer 2012] and [de Lima Bezerra 2011]. Because the Iterative algorithm did not reported interesting results, we decided to withdraw it of analysis. Actually, we are interested to know if the performance of *Dynamic Threshold Algorithm* is better than: (i) Threshold Algorithm, which follows similar ideas; (ii) Sampling Algorithm, which presented best performance in previous comparisons [Bezerra and Wainer 2012; de Lima Bezerra 2011]; (iii) and Naive Algorithm, which has higher *recall*, but lower *precision*.

In the statistical analysis we applied One-way ANOVA, for identifying if there is at least one algorithm with a significant difference of performance among others, followed by Tukey HSD, for identifying pair-to-pair the best results[Steinberg 2010]. Finally, we use a ROC Graph for graphically presenting the performance difference between the two best results (See Figure 1). In such a graph, the perfect classifier would appropriately classify all the positive occurrences (TPR = 100%), and it would not misclassify as positive the normal occurrences (FPR = 0%). In addition, a good classifier



(a) Complete Logs.

(b) Simpler Logs.

Fig. 1: ROC Graphs. Comparison between Sampling and Dynamic Threshold Algorithms

is plotted above the main diagonal. For example, the Naive Algorithm has 100% of TPR, but it also has a 100% of FPR, so it would be plotted over the main diagonal.

6.1 First Set of Logs - Complete Logs

Table II: Performance Mean of Complete Logs - Accuracy, f1-measure and f4-measure

	ACC	F-Measure	F4-Measure
DYNAMIC THRESHOLD	**0.6595**	**0.6165**	0.8054
SAMPLING	0.4665	0.5614	**0.8863**
THRESHOLD	0.4325	0.1808	0.3048
NAIVE	0.3285	0.4735	0.8659

We report in Table II the performance mean of algorithms over the complete logs (first dataset). Considering accuracy and f1-measure, *Dynamic Threshold Algorithm* had best performance, while for f4-measure, Sampling Algorithm had best performance.

It is worth notice that statistically, with 95% of significance: (i) the difference in the accuracy between Dynamic Threshold and Sampling algorithms is significant (p-value is almost zero); (ii) the difference in the f4-measure between Dynamic Threshold and Sampling algorithms is significant (p-value is 0.0013007); (iii) the difference in the f1-measure between Dynamic Threshold and Sampling algorithms is statically significant (p-value is 0.0345066); (iv) finally, we show in Figure 1a a ROC Graph comparing the best two classifiers, and the *Dynamic Threshold Algorithm* is selected as the best classifier.

6.2 Second Set of Logs - Simpler Logs

We report in Table III the performance mean of algorithms over the simpler logs (second dataset). Considering accuracy and f1-measure, *Dynamic Threshold Algorithm* had best performance, while for

Table III: Performance Mean of Simpler Logs - Accuracy, f1-measure and f4-measure.

Algorithm	ACC	F1-Measure	F4-Measure
DYNAMIC THRESHOLD	**0.8147303968**	0.7532466836	0.8948376809
SAMPLING	0.7870617918	0.7526751987	**0.9407704653**
THRESHOLD	0.3790339921	0.3219208585	0.5512582595
NAIVE	0.3509796491	0.4742572677	0.8064932184

f4-measure, Sampling Algorithm had best performance.

It is worth notice that statistically, with 95% of significance: (i) the difference in the accuracy between Dynamic Threshold and Sampling algorithms is significant (p-value is 0.0239637); (ii) the difference in the f4-measure between Dynamic Threshold and Sampling algorithms is also significant (p-value is 3e-07); (iii) the difference in the f1-measure between Dynamic Threshold and Sampling algorithms *is not* significant (p-value is 0.9997922); (iv) finally, we show in Figure 1b a ROC Graph comparing the best two classifiers, and the *Dynamic Threshold Algorithm* is selected as the best classifier – in the simpler dataset, Dynamic Threshold Algorithm had a performance closer to Sampling Algorithm.

7. CONCLUSION AND FUTURE WORK

Because detection of anomalies are a growing research area, data mining community has published a large and growing body of literature in the subject. It has been realized that fraud and intrusion detection in systems, novelty detection in time series, bio-threats discovery, and other similar tasks are better understood as anomaly detection[Chandola et al. 2009]. Besides, a huge number of systems, and therefore logs are available nowadays. This is a result of the interest of companies for supporting the control of their businesses processes, for the coordination and control of the businesses supported by these tools guaranties that the execution of activities obey its prescribed definition. On the other hand, the companies that demand a flexible automation of their business processes, a rigid control may compromise their response to new market strategies. Nevertheless, a flexible system may be vulnerable to fraudulent or undesirable executions.

In order to provide a solution to balance the trade off between flexibility and security, we presented in this work a new algorithm, the *Dynamic Threshold Algorithm*, for anomaly detection of traces in logs of PAS. The algorithm was assessed through two datasets of logs, one with *360 complex logs* and other with *1800 simpler logs*. The simpler log have shorter traces and none of the traces repetition of activities (which means that their correspondent process models have no loops). The *Dynamic Threshold Algorithm* has a statistically significant better accuracy for both dataset of logs against the algorithms proposed in [Bezerra and Wainer 2012]. Besides, when considering the f1-measure, the new algorithm also proved to be better.

However, because the emphasis of this work is on anomalies as frauds, we would choose f4-measure as the most appropriate metric for selecting the best algorithm, for, as we stated in Section 5, such an adoption increases the importance of the recall value. In this case, the *Sampling Algorithm* has a statistically significant better performance. Moreover, when comparing the new anomaly detection approach with the Threshold Algorithm, which also classifies based on a threshold value, we can conclude that the Dynamic Threshold Algorithm had better performance, in all metrics (see Tables II and III).

Those results are important to point out different characteristics of both Sampling and Dynamic Threshold Algorithms. *Sampling Algorithm* is better at finding the anomalies in the logs, but pays the cost of a higher false positive rate, while *Dynamic Threshold* has better accuracy, and a good balance between precision and recall (f1-measure). Therefore, as future work, we believe that one could combine the results of both algorithms into a single decision. We are currently exploring some alternatives in this direction.

We are also currently exploring the use of other perspectives of process mining in anomaly detection. In real life, anomalies can be limited not only to the control-flow perspective, but also to the data, time and organizational perspectives. For example, a particular execution path may be used only when executed by senior roles, so a particular trace that followed that path is not in itself anomalous, neither are traces whose activities were executed by junior roles. But a trace that followed that path and had activities executed by a junior role are anomalous, and this determination involved both the control-flow and the organizational flow perspectives taken together.

Finally, the models and the logs used in this article are available online[3], so that researchers can propose new algorithms and fairly compare them to ours, and future users of our algorithms may evaluate if our models and logs are representative of the models and logs they will face in their problems.

REFERENCES

Aalst, W., Adriansyah, A., and Medeiros, A. Process mining manifesto. *Business Process* 99 (2): 169–194, 2012.

Agarwal, D. K. An empirical bayes approach to detect anomalies in dynamic multidimensional arrays. In *ICDM*. Texas, USA, pp. 26–33, 2005.

Agrawal, R., Gunopulos, D., and Leymann, F. Mining process models from workflow logs. In *EDBT '98: Proceedings of the 6th International Conference on Extending Database Technology*. Lecture Notes in Computer Science, vol. 1377. Springer, Valencia, Spain, pp. 469–483, 1998.

Bezerra, F. and Wainer, J. Anomaly detection algorithms in business process logs. In *10th International Conference on Enterprise Information Systems*. SciTePress, Barcelona, Spain, pp. 11–18, 2008a.

Bezerra, F. and Wainer, J. Anomaly detection algorithms in logs of process aware systems. In *SAC '08: Proceedings of the ACM Symposium on Applied Computing*. ACM, Fortaleza, Ceara, Brazil, pp. 951–952, 2008b.

Bezerra, F. and Wainer, J. Fraud detection in process aware systems. *International Journal of Business Process Integration and Management (IJBPIM)* 5 (2): 121–129, 2011.

Bezerra, F. and Wainer, J. Algorithms for anomaly detection of traces in logs of process aware information systems. *Information Systems* - (-): In Press, 2012.

Bezerra, F., Wainer, J., and Aalst, W. M. P. Anomaly detection using process mining. In *Enterprise, Business-Process and Information Systems Modeling*. Lecture Notes in Business Information Processing, vol. 29. Springer, Amsterdam, The Netherlands, pp. 149–161, 2009.

Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* 41 (3): 1–58, 2009.

Cook, J. E., Du, Z., Liu, C., and Wolf, A. L. Discovering models of behavior for concurrent workflows. *Computers in Industry* 53 (3): 297–319, 2004.

Cook, J. E. and Wolf, A. L. Discovering models of software processes from event-based data. *ACM Trans. Softw. Eng. Methodol.* 7 (3): 215–249, 1998.

de Lima Bezerra, F. *Algoritmos de Detecï£¡iï£¡o de Anomalias em Logs de Sistemas Baseados em Processos de Negï£¡cios*. Ph.D. thesis, Universidade Estadual de Campinas, Campinas, Sï£¡o Paulo, 2011. In Portuguese.

de Medeiros, A., van der Aalst, W., and Weijters, A. Workflow mining: Current status and future directions. In *On The Move to Meaningful Internet Systems: CoopIS, DOA, and ODBASE*. LNCS, vol. 2888. Springer, Sicily, Italy, pp. 389–406, 2003.

de Medeiros, A. K. A. *Genetic Process Mining*. Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2006.

de Medeiros, A. K. A., Weijters, A. J. M. M., and van der Aalst, W. M. P. Genetic process mining: A basic approach and its challenges. In *Business Process Management Workshops*. LNCS, vol. 3812. Springer, Nancy, France, pp. 203–215, 2006.

Donoho, S. Early detection of insider trading in option markets. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '04. ACM Press, New York, USA, pp. 420–429, 2004.

Dumas, M., van der Aalst, W., and ter Hofstede, A. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley, 2005. ISBN 13 978-0-471-66306-5.

Fawcett, T. and Provost, F. Adaptive fraud detection. *Data Mining and Knowledge Discovery* 1 (3): 291–316, January, 1997.

[3] `http://www.fabiobezerra.pro.br/logs/`

HAMMORI, M., HERBST, J., AND KLEINER, N. Interactive workflow mining - requirements, concepts and implementation. *Data & Knowledge Engineering* 56 (1): 41–63, 2006.

HERBST, J. AND KARAGIANNIS, D. Workflow mining with inwolve. *Computers in Industry* 53 (3): 245–264, 2004.

LARSON, R. AND FARBER, E. *Elementary statistics: picturing the world.* Prentice Hall, 2003.

LEE, W. AND XIANG, D. Information-theoretic measures for anomaly detection. In *IEEE Symposium on Security and Privacy.* Oakland, California, pp. 130–143, 2001.

NOBLE, C. C. AND COOK, D. J. Graph-based anomaly detection. In *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge discovery and data mining.* ACM Press, New York, USA, pp. 631–636, 2003.

PANDIT, S., CHAU, D. H., WANG, S., AND FALOUTSOS, C. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *WWW '07: Proceedings of the 16th international conference on World Wide Web.* ACM Press, New York, USA, pp. 201–210, 2007.

PINTER, S. S. AND GOLANI, M. Discovering workflow models from activities' lifespans. *Computers in Industry* 53 (3): 283–296, 2004.

ROZINAT, A., DE MEDEIROS, A. A., GÜNTHER, C., WEIJTERS, A., AND VAN DER AALST, W. Towards an evaluating framework for process mining algorithms. Tech. rep., Technische Universiteit Eindhoven, 2007.

ROZINAT, A. AND VAN DER AALST, W. Conformance checking of processes based on monitoring real behavior. *Information Systems* 33 (1): 64–95, March, 2008.

ROZINAT, A. AND VAN DER AALST, W. M. P. Conformance testing: Measuring the fit and appropriateness of event logs and process models. In *Business Process Management Workshops.* LNCS, vol. 3812. Springer, Nancy, France, pp. 163–176, 2005.

SABHNANI, R., NEILL, D., AND MOORE, A. Detecting anomalous patterns in pharmacy retail data. In *Proceedings of the KDD 2005 Workshop on Data Mining Methods for Anomaly Detection.* ACM, Chicago, USA, 2005.

SCHIMM, G. Mining exact models of concurrent workflows. *Computers in Industry* 53 (3): 265–281, 2004.

STEINBERG, W. *Statistics Alive!* SAGE Publications, 2010.

VAN AALST, W. M., VAN HEE, K. M., VAN WERF, J. M., AND VERDONK, M. Auditing 2.0: Using process mining to support tomorrow's auditor. *Computer* 43 (3): 90–93, March, 2010.

VAN DER AALST, W. M. P. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer, 2011.

VAN DER AALST, W. M. P. AND DE MEDEIROS, A. K. A. Process mining and security: Detecting anomalous process executions and checking process conformance. *Electronic Notes in Theoretical Computer Science* 121 (4): 3–21, February, 2005.

VAN DER AALST, W. M. P., VAN DONGEN, B. F., HERBST, J., MARUSTER, L., SCHIMM, G., AND WEIJTERS, A. J. M. M. Workflow mining: A survey of issues and approaches. *Data & Knowledge Engineering* 47 (2): 237–267, 2003.

VAN DER AALST, W. M. P. AND WEIJTERS, A. J. M. M. Process mining: a research agenda. *Computers in Industry* 53 (3): 231–244, 2004.

VAN DER AALST, W. M. P., WEIJTERS, T., AND MARUSTER, L. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering* 16 (9): 1128–1142, 2004.

VAN DONGEN, B., DE MEDEIROS, A., VERBEEK, H., WEIJTERS, A., AND VAN DER AALST, W. The ProM Framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets.* LNCS, vol. 3536. Springer, Miami, USA, pp. 444–454, 2005.

VAN DONGEN, B. AND VAN DER AALST, W. Multi-phase process mining: Building instance graphs. In *Conceptual Modeling - ER 2004.* Lecture Notes in Computer Science, vol. 3288. Springer Berlin / Heidelberg, Shanghai, China, pp. 362–376, 2004.

VAN RIJSBERGEN, C. J. *Information Retrieval.* Butterworths, 1979.

WAINER, J., KIM, K., AND ELLIS, C. A. A workflow mining method through model rewriting. In *Groupware: Design, Implementation, and Use: 11th International Workshop.* LNCS, vol. 3706. Springer, Porto de Galinhas, Brazil, pp. 184–191, 2005.

WEIJTERS, A. AND VAN DER AALST, W. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering* 10 (2): 151–162, 2003.

WEIJTERS, A., VAN DER AALST, W., AND DE MEDEIROS;, A. A. Process mining with the heuristicsminer algorithm. Tech. rep., Beta Research School for Operations Management and Logistics, 2006.

WEN, L., WANG, J., AND SUN, J. Detecting implicit dependencies between tasks from event logs. In *Frontiers of WWW Research and Development - APWeb 2006.* LNCS, vol. 3841. Springer, Harbin, China, pp. 591–603, 2006.

YANG, W.-S. AND HWANG, S.-Y. A process-mining framework for the detection of healthcare fraud and abuse. *Expert Systems with Applications* 31 (1): 56–68, July, 2006.