Fraud detection in process aware systems

Fábio Bezerra* and Jacques Wainer

Institute of Computing – UNICAMP,
Av. Albert Einstein, 1251, Campinas, São Paulo, Brazil
E-mail: fbezerra@ic.unicamp.br
E-mail: wainer@ic.unicamp.br
*Corresponding author

Abstract: In the last years, some large companies have been involved in scandals related to financial mismanagement, which represented a large financial damage to their stockholders. To recover market confidence, certifications for best practices of governance were developed, and in some cases, harder laws were implemented. Companies adhered to these changes as a response to the market, deploying process aware systems (PAS) and adopting the best practices of governance. However, companies demand a rapid response to strategic changes or changes in business models between partners, which may impose serious drawbacks to the adoption of normative PAS to the competitiveness of these companies. Thus, while companies need flexible PAS, flexibility may compromise security. To re-balance the trade-off between security and flexibility, we present in this work an anomaly detection algorithm for PAS. The identification of anomalous events can help the adoption of flexible PAS without the loss of security properties.

Keywords: anomaly detection; process aware systems; PAS; process mining; flexible workflow systems.

Reference to this paper should be made as follows: Bezerra, F. and Wainer, J. (2011) 'Fraud detection in process aware systems', *Int. J. Business Process Integration and Management*, Vol. 5, No. 2, pp.121–129.

Biographical notes: Fábio Bezerra is a PhD student of Institute of Computing at State University of Campinas (IC-UNICAMP), in São Paulo, Brazil. He is supervised by Prof. Jacques Wainer since 2006, when he started his PhD course. His PhD course has been researching in process mining field and anomaly detection in logs of process aware systems. Also, he is interested in business process management and flexible workflow management systems.

Jacques Wainer graduated with a degree in Electronic Engineering from the University of São Paulo (1982) and PhD in Computer Science from the Department of Computer Science – Pennsylvania State University (1991). He is an Associate Professor of the Institute of Computing in the State University of Campinas (UNICAMP). He has experience in computer science with emphasis in artificial intelligence, collaborative systems, medical informatics and impacts of information technology in society.

This paper is a revised and expanded version of a paper entitled 'Fraud detection in process aware systems' presented at the II Workshop on Business Process Management, Vila Velha, Brazil, 26–29 October 2008.

1 Introduction

The Sarbanes-Oxley Act, also known as *SOx* or *Sarbox*, is a USA Federal Law enacted in response to a number of major corporate and accounting scandals (e.g., Enron and WorldCom). These scandals, which cost investors billions of dollars when the share prices of the affected companies collapsed, shook public confidence in the nation's securities markets. The *SOx* contains 11 titles that describe specific mandates and requirements for financial reporting, which imposes to major companies a redefinition of their operational processes in order to comply with new law. This scenario motivated the companies to adhere a set of best practices of governance – e.g., COSO, Committee of Sponsoring Organizations of the Treadway Commission, and COBIT, control objectives for information and related technology. Moreover, there are countries that the adherence of best practices of governance may indicate a competitive factor. For example, in Brazil the adherence of companies to the different levels of compliance to corporate governance is not a constraint law, but it is a competitive differential in the stock market. Thus, the companies became an interesting environment to adopt process aware information systems – workflow management systems (WfMS), enterprise resource planning (ERP), supply chain management (SCM), etc. – to control their business processes, so they would be able to present reliable financial reports in a short time.

Despite the automation provided by process aware systems, the business process control of competitive companies can not be supported by a normative tool like aWfMS, because such companies demand a flexible automation of their business processes. This flexibility is necessary because the companies need to respond rapidly to new market strategies or new business models, and process aware systems with too much operational constraint may not provide the appropriate environment to deploy such business changes. On the other hand, a flexible process aware system is vulnerable to fraudulent or undesirable executions, which clearly imposes a trade-off between flexibility and security. In other words, the system should provide flexibility for competitiveness reasons, but it also should avoid or *identify* misuse of system.

Thus, in order to fit flexibility and security in the same system, it is important to develop methods that can detect anomalous instances of enacted business processes, that is, methods that can identify fraud attempts. We believe that the coexistence of security and flexibility requirements may be provided into a PAS through the aggregation of a tool that can dynamically detect fraud executions in the log of these systems.

By and large, a log generated by such systems is comprised of process instances, referred in this work as traces, which are a stream view of process instances. For example, a trace *abc* means that the task *a* was concluded before task *b*, and task *b* was concluded before task *c*. A common intuition is to consider a fraudulent trace an infrequent or rare event in the log. Nevertheless, this paper argues that although an anomalous trace is an infrequent event, an infrequent trace not necessarily indicates an anomaly, as will be shown later. This premise imposes a challenge: What infrequent trace could be classified as anomalous? This work will present an algorithm that collaborates in this mission, and it will assess the effectiveness of the proposed anomaly detection solution.

In order to define an anomaly detection tool, we are interested in applying process mining techniques to discover anomalies or outliers in the log. Although *process mining* algorithms were created to be used to discover or mine process models from logs of PAS (Agrawal et al., 1998; Maruster et al., 2001; de Medeiros et al., 2003; van der Aalst et al., 2004), there are some academic results that provide applicability in anomaly detection field, as in van der Aalst and de Medeiros (2005), and Bezerra and Wainer (2007a, 2007b; 2008a, 2007b). Nevertheless, considering the anomaly detection efficacy of these algorithms, it is interesting to implement extension approaches that consider other process mining algorithms or 'noise' metrics.

In those works, the classification of an anomalous trace is based on the 'noise' which a trace makes in a process model, which was discovered by a process mining algorithm. These algorithms are based on the rationale that if a trace is not an instance of a process model, then the model will require some structural changes to fit the trace; moreover, in the case of an anomalous trace, the structural changes will be probably higher.

This work is closely related to the work presented in Bezerra and Wainer (2007a), but it differs because it defines a new 'noisy' metric, uses a new threshold value in the algorithm, and also evaluates the metrics with a different and bigger set of logs. Thus, in order to assess these differences this work is organised as follows. Section 2 reports related work in process mining and anomaly detection field. In Section 3, we describe the anomaly detection algorithm and some extension points of that algorithm, while in Section 4 we present the evaluation of the algorithm. Finally, in Section 5 we conclude our work, and we point out some ideas for future work.

2 Related work

The large adoption of PAS by companies, both to better control their business processes and desire to improve the confidence of public, developed an interesting scenario to apply business intelligence techniques over the log generated by these systems. For example, process mining techniques allow for various types of analysis based on so-called event logs. Using process mining one can reconstruct a process model from a log generated by some information system. In the last ten years, researchers around the world have been working on such techniques (van der Aalst et al., 2003, 2004; de Medeiros et al., 2003). The term was first coined in the context of software processes. 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 mining, the paper of Agrawal et al. (1998), present an algorithm that mine models having three properties in mind: completeness, minimality, and irredundancy.

Moreover, new process mining approaches have been proposed recently (van der Aalst et al., 2003, 2004; van der Aalst and Weijters, 2004; Schimm, 2004; Hammori et al., 2006; de Medeiros et al., 2006). Hammori et al. (2006) present an interactive approach to mine process models because it considers the setting of some parameters by an analyst. Among the recent process mining approaches, the most visible one is the α–algorithm (van der Aalst et al., 2003, 2004; van der Aalst and Weijters, 2004). The effectiveness of that algorithm was formally proved for a class of process models, the workflow net (WF-Nets), which are Petri nets that require:

1 a single start place

2 a single end place

3 every node must be on some path from start to end.

However, such algorithms have severe limitations, e.g., the inability to deal with short loops. Schimm (2004) presents a mining approach that also discovers a model after merging or rewriting the traces of log with a set of formal rules (axioms) from a workflow algebra. Besides, different from more disseminated approaches, the output model presented by Schimm is a block-structured workflow model. Similar to Schim's approach, Wainer et al. (2005), argue that the process mining problem is not well defined because many solutions with 100% of fitness can be inferred, and present a sketch of a process mining algorithm that characterises the stated problem. That algorithm is an incremental mining 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 problem that considers the selection of best model for a given log.

In this work, because the classification process of a trace is based on the assessment of a model dynamically discovered, we apply process mining techniques to support the construction of such models dynamically discovered. In Bezerra and Wainer (2008a, 2008b), we highlight in conclusion that the accuracy of anomaly detection is related to the efficacy of process mining result.

The problem of dealing with noise in the event log is closely related to anomaly detection field, and there are some process mining methods that deal with the mining of noisy logs (Agrawal et al., 1998; van der Aalst et al., 2003; Cook et al., 2004; Pinter and Golani, 2004; Herbst and Karagiannis, 2004), yet their approaches are limited to the frequency evaluation of dependency relation between two activities. For example, infrequent dependency relations between two activities may not be modelled in the resulting process model. A more sophisticated and promising approach, called genetic mining, was proposed in de Medeiros et al. (2006), and de Medeiros (2006). This algorithm is based on genetic algorithms, which search for a solution (an individual) that satisfies a selection criteria, called fitness function. The individuals are generated based on genetic operators such as crossover, mutation, and elitism.

In addition to process mining area, this work is related to data mining field, which is also interested in anomaly detection area. Such an area has been applied in different application fields, and it has received a special attention of data mining community. For instance, 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 (e.g., Lee and Xiang, 2001; Noble and Cook, 2003). Other efforts are concerned with the detection of fraudsters in auctions or e-commerce sites (e.g., Pandit et al., 2007).

As stated before, there are many solutions related to anomaly detection field, but in the context of PAS there are few solutions. The work of van der Aalst and de Medeiros (2005) is closely related to our work. In that work, the authors present two anomaly detection methods that are supported by α-algorithm. However, that algorithm has a serious limitation because it demands the existence of a known 'normal' log, which is used to mine a process model that will classify the traces of an audit log. That is, in application domains that demand a flexible support the existence of a known 'normal' log may not be available. In Yang and Hwang (2006), the authors present a framework to detect fraud and abuse in health insurance systems. In this 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.

Also related to the context of PAS, Bezerra and Wainer (2007a, 2007b; 2008a, 2008b) present some anomaly detection approaches. In those papers, the authors cite three extension points that may influence the accuracy of the anomaly detection algorithm: process mining algorithm, noise metric, and size of sampling. Because this work also uses the conformance variance to classify a trace as an anomalous instance, it is closely related to the work in Bezerra and Wainer (2007a), but we can point out two major differences:

1. this work uses a new noise metric that evaluates the complexity of a model, which is used to classify a trace as anomalous or normal
2. this work redoes the assessment of algorithms with a different set of logs, supporting the results of previous work.

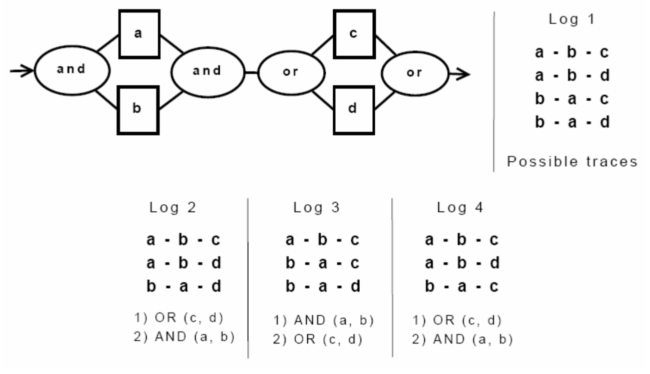
Therefore, considering the conformance assessment of a model regarding a log, this work is closely related to the work in Rozinat and van der Aalst (2005, 2008), which present different dimensions for the evaluation of process models. Beyond fitness dimension, which describes how much of log can be correctly played in a model; the authors also present the precision and structure dimensions. Precision dimension indicates how much behaviour is supported by a model, whereas structure dimension indicates how much complex is a model. In Rozinat et al. (2007), the authors present a more complete analysis of different assessment metrics, comparing different evaluation aspects of each metric (e.g., input, output range, dimension of assessment, and computational complexity)

3 Anomaly detection algorithms

The concept of anomaly in a log can be related to different semantics. For instance, an anomaly can be a noise when an event (task) is not registered into the log, or it is registered in duplicity, consequence of a log component error. In addition, an anomaly can represent an exception, an operational error, or a fraud attempt. An exception characterises an abnormal or unusual execution, but it can be supported by the business. For example, in a healthcare system there are a lot of patient paths that deviate from standard guidelines. As another example, a fraud attempt and an operational error are unusual executions that provoke undesirable results to the business.

Independently of semantic, it is common to consider an anomaly as a rare or infrequent event. However, it is a naive approach to classify a trace considering only its frequency in the log. For example, is it appropriate to classify as anomalous the trace whose frequency is inferior to 4%? Some paths in the process model can be enacted more frequently than others, so probably some 'normal' traces are infrequent. For that reason, we do not believe in an anomaly detection method based *only* on the frequency of traces. Figure 1 illustrates this problem. It depicts a process model and four logs – a log with all possible traces of the model (Log 1) and three logs (Log 2, Log 3, and Log 4) that represent subsets of all possible traces. As it is possible to notice, each trace of four logs can be played by the depicted model.

Figure 1 Mining with a subset of possible *traces*



Notes: Examples of incomplete logs. The semantic blocks induced by mining are bellow of each incomplete log.

In this example, all incomplete logs can generate, by process mining, the same model. For example, in Log 2, after mining the traces [$a - b - c$] and [$a - b - d$], a model [$a - b -$ or (c, d)] can be inferred, representing a selection between activities c and d ($OR(c, d)$). Then, adding the trace [$b - a - d$] to the model [$a - b - or(c, d)$], a new model [$and(a, b) - or(c, d)$] can be inferred, representing the parallelism between activities a and b. In this example, the Log 2 has not the trace [$b - a - c$], but it can be created by the mined model [$and(a, b) - or(c, d)$] because there is a path in model that fits with the trace [$b - a - c$]. Therefore, even if the trace [$b - a - c$] was infrequent in the log, it would not be classified as anomalous.

3.1 Algorithm: preliminary definition

As stated before, an infrequent trace can be an anomaly or a normality. However, we can assume that a frequent trace represents a normal execution, otherwise its repetitions would be easily noticed by a business process controller, or it would not be allowed. Because there is not a clear definition of what it is an anomalous trace, in this work, we adopt the following rationale to help us detect anomalous traces:

> "Given a process model mined with almost all traces from a log, the compliance level between this model and the whole log is inferior when the whole log comprises an anomalous trace."

Therefore, the proposed method will classify as anomalous traces the traces whose compliance level variance is greater than a given threshold value. This definition is preliminarily presented in Algorithm 1.

Algorithm 1 Preliminary definition of anomaly detection algorithm

```
Input: A log $L$, which is a set of traces generated by a PAS.
Output: A set of traces $A$ that was classified as anomalous traces.
AnomalousTraces($L$)
(1)  $T_f$ is a set of frequent traces from log $L$;
(2)  $T = \{\text{different traces of } L\} - T_f$;
(3)  for each race $t \in T$, evaluate the compliance variance
(4)      $L' = L - \{t\}$;
(5)      Evaluate the conformance value $c_1$ for the log without the trace $t$;
(6)      Evaluate the conformance value $c_2$ for the log $L$;
(7)      Evaluate the conformance variance $c = | c_1 - c_2 |$;
(8)      Add the tuple $(t, c)$ in the set $C = \{(t, c) \in T \times \Re\}$;
(9)  $A$ is the set of anomalous traces (initially empty);
(10) for each $(t, c) \in C$
(11)     if $c \geq threshold$
(12)         Add $t$ in the set $A$;
(13) return $A$
```

In the first step of the algorithm, a set of frequent traces are identified and separated. In order to classify a trace as frequent or infrequent, we adopt a heuristic value of 10%, so we say that a trace is frequent if its frequency is at least 10%. Although this is a *heuristic* value, we believe that 10% seems to be reasonable to represent those process paths that are more excited in real scenarios. Moreover, it is important to cite that such a step is used for optimisation reasons, since it complies with our premise that frequent traces would not be anomalies. Thus, the second step is used to compose a set of candidate anomalous traces, that is, the infrequent traces.

This preliminary definition of Algorithm 1 is not complete, since we need to define the value of *threshold* variable. Therefore, in order for this algorithm to work, we need to discover how much is the compliance variance level when the log comprises only 'normal' traces, that is, traces which could be an instance of a known model. Then, the analysis of compliance variance values in 'normal' scenarios would help us infer a *threshold* variance of conformance metric for 'normal' traces. After this investigation, we can consider that anomalous traces are traces that have a compliance variance greater than the variance in 'normal' scenarios.

In Section 3.2, we will discuss the conformance metrics that will be used in the complete definition of algorithms, and we will describe the analysis carried out to define the corresponding threshold values.

3.2 Study of some metrics

This study was carried out for each different trace of a log, and it will try to define a threshold value that best describes the conformance variance that a trace may provoke, in scenarios of logs without occurrences of anomalous traces. The definition of this threshold value works as a delimiter (or classifier) of 'normal' and 'anomalous' traces. In other words, as stated in the preliminary definition in Section 3.1, we assume that anomalous traces are traces that will give us a compliance or conformance variance value greater than a threshold value.

To quantify the compliance of a log and a model, we used the conformance metrics presented in Rozinat and van der Aalst (2005). These metrics are based on two dimensions: fitness and appropriateness. The *fitness* dimension is measured by a metric with the same name. While the appropriateness dimension is measured by two metrics, *structural* and *behavioural* appropriateness. These metrics are real numbers between zero and one, where the value zero means that there is not a fitness or an appropriateness between the model and the log, whereas, the value one means that there is a total compliance between the model and the log. For instance, the fitness metric reports a value which indicates how compliant is a model and a log. If the model is 100% compliant its fitness is 1 and it indicates that all traces can be played over the model. On the other hand, if the model is totally incompliant its fitness is 0. The structural and behavioural appropriateness metrics report the preference between two fitted models. The structural appropriateness penalises bigger or much specific models, while the behavioural appropriateness penalises much generic models. Therefore, the structural metric is applied to a model, whereas the behavioural metric, similar to fitness metric, is applied to a pair of model and log.

Besides, we defined a new metric called size metric that is similar to structural metric because it also considers the complexity of the process model. It is a metric that represents the counting of places, transitions and edges in a Petri net. Moreover, it is not a conformance metric because it does not evaluate the correspondence between a log and a model, but it assesses only the complexity (number of elements) of process model. This metric was defined and used in this study because we believe that a log with anomalous traces induces a process model that is more complex than a model induced by the same log without anomalous traces. That is, we believe that a model mined with normal and anomalous traces will have more paths than a model mined without anomalous traces.

The study of these metrics was carried out with 149 logs without anomalous traces. We consider that a log does not have anomalous traces because it is filled by traces created from a known process model. Thus, the creation of a log is based on a model randomly created by a function that utilises four parameters, as follows:

1 the maximum length of a trace

2 the minimum number of different traces that a model can instance

3 the maximum number of different traces that a model can instance

4 the number of output models.

A log is created through playing all paths of a model. Then, the traces created after playing the model are used to fulfil the log, but with a non-uniform frequency – some paths are more excited than others. Figure 1 can be used again like an example because it illustrates a process model and its four respective traces. In the context of this study, each log contains between 4 and 15 different classes of traces, whose length is between 4 and 7. In addition, each log was fulfilled with 100 traces. Thus, for each log L from these 149 logs, we collect the conformance variance for each class of trace $t \in L$ as follows:

1 a model M_r was created with the *remaining* traces $(L - t)$

2 a tuple $C_1(F_1, S_1, B_1)$ was created where:
   - F_1 is the fitness of M_r and $L - t$
   - S_1 is the structural appropriateness of M_r
   - B_1 is the behavioural appropriateness of M_r and $L - t$
   - X_1 is the size of M_r

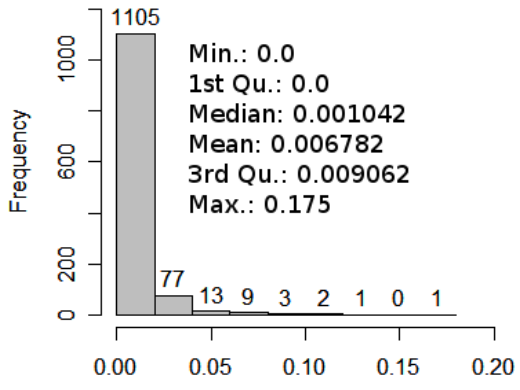
3 a model M_a was created with all traces (L)

4 a tuple $C_2(F_2, S_2, B_2)$ was created where:
   - F_2 is the fitness of M_r and L
   - S_2 is the structural appropriateness of M_a (model created from all traces)
   - B_2 is the behavioural appropriateness of M_r and L
   - X_2 is the size of M_a (model created from all traces)

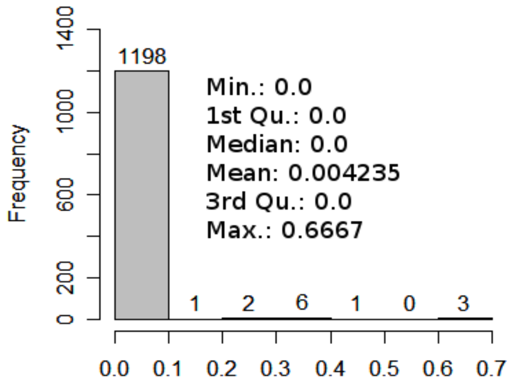
5 a variance tuple $C_v(F_v, S_v, B_v, X_v)$ was created and saved where:
   - $F_v = |F_1 - F_2|$
   - $S_v = |S_1 - S_2|$
   - $B_v = |B_1 - B_2|$
   - $X_v = |X_1 - X_2| / X_1$.

The models M_r and M_a were created through the α-algorithm, a process mining algorithm which generates a Petri net representation of a process model van der Aalst et al. (2004). Each tuple saved in step 5 represents a record of a table of conformance variance values (in statistical meaning, individuals), whose values were assessed, as it is possible to see in Figure 2. This figure comprises three histograms and a corresponding statistical report for each metric, where y-axis reports the frequency, and x-axis reports the conformance variance value. In the case of size metric, the variance was normalised by the first size value, which avoids a misinterpretation for a same size variance, but two different original models with different sizes. Moreover, the figure does not report structural variance analysis because all collected variance values for this metric was zero.

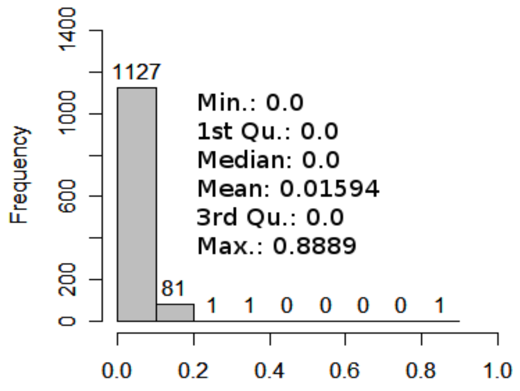
Figure 2 Histogram and statistical analysis for study of metrics, (a) histogram of fitness variance (b) histogram of behavioural variance (c) histogram of size variance



(a)



(b)



(c)

It is worth notice that those graphics report our intuition that for 'normal' traces the majority of values represent minimum variation in the conformance metric. For instance, 1,105 individuals have a fitness variance between 0.00 and 0.02, 1198 individuals have a behavioural variance between 0.00 and 0.1, and 1208 individuals have a normalised size variance between 0.00 and 0.2.

3.3 *Extension approaches from preliminary definition*

The preliminary definition of algorithm considers the evaluation of two conformance values, and its variance. Right now, we will present four different metrics that will complete the definition of algorithm. Each metric will represent an extension of algorithm. Moreover, all the extensions consider the same threshold value, which was induced by first quartile (1st Q. = 0.0) of prior study. For instance, the structural variance is zero for at least 75% of experiments (3rd quartile is zero for structural metric), and a trace with a structural variance greater than zero is a good candidate to be an anomalous one.

In Algorithm 2, we present an extension based on size metric as a complete definition of algorithm. The same rationale would be applied for other extensions, but replacing size metric by other metrics (fitness, structural appropriateness, and behavioural appropriateness).

Algorithm 2 Fraud detection algorithm based on size metric

```
Input: A log $L$, which is a set of traces generated by a PAS.
Output: A set of traces $A$ that was classified as anomalous traces.
AnomalousTraces($L$)
(1)   $T_f$ is a set of frequent traces from log $L$;
(2)   $T$ = {different traces of $L$} – $T_f$;
(3)   for each race $t \in T$, evaluate the compliance variance
(4)       $L' = L - \{t\}$;
(5)       Evaluate the size value $c_1$ for the log without the trace $t$;
(6)       Evaluate the size value $c_2$ for the log $L$;
(7)       Evaluate the (normalised) size variance $c = | c_1 - c_2 | / c_1$;
(8)       Add the tuple $(t, c)$ in the set $C = \{(t, c) \in T \times \Re\}$;
(9)   $A$ is the set of anomalous traces (initially empty);
(10)  for each $(t, c) \in C$
(11)      if $c > 0$
(12)          Add $t$ in the set $A$;
(13)  return $A$
```

4 Assessment

We have assessed the algorithms with a set of synthetic logs which have been created based on the traces of process models dynamically created. Two reasons influenced our choice for synthetic data. 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. Last but not least, a real log is not easily available. Therefore, as we know the process model that was used to create a log ('normal traces'), it is easy to identify the anomalous traces in the log since an anomalous trace is a trace that is not an instance of a known model, that is, a trace that can not be played by model.

4.1 Methodology

The experiments were based on 435 logs with different configurations. Initially, we randomly created 145 process models which were the matrix of logs. Such models could instance at least eight unique traces and at most 15 unique traces. For each model, we instanced 80 traces that were used to compose the log and represent the normal traces to the experiment. In addition, the unique traces of each log were added in a non-uniform frequency distribution. After that, we merged some anomalous traces to define six types of log, as follows:

1 a *log A* that has one single anomalous trace

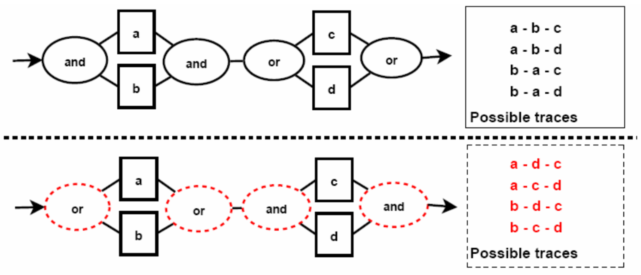
2 a *log B* that has two single anomalous traces

3 a *log C* that has three single anomalous traces.

With regard to the anomalous traces, they were generated as instances of a process model created after shifting AND-Blocks to OR-Blocks, and vice versa. Such a process model is known only during log generation, but it is unknown during anomaly detection assessment. The instances of this new (shifted) process model are used to define the anomalous traces for the original model. Then, after an instance has been created some events could be removed of the trace, or two random events could be interchanged.

Figure 3 illustrates an example of definition of anomalous traces. The possible traces of upper model are anomalous for the lower model, and vice-versa. In this example, while the tasks *c* and *d* could not play together in the upper model, they are played together in the lower model. On the other hand, while the tasks *a* and *b* would have to play together in the upper model, they are not played together in the lower model.

Figure 3 Example of definition of anomalous traces (see online version for colours)



The approach used to create the anomalous traces represents our intuition that anomalous traces are similar to 'normal' traces, for they are based on the same set of tasks. In other words, we believe that in real scenarios a fraudster will not attempt to execute new tasks, but he will try to make 'little changes' in a standard operational procedure, because it will be more difficult that his fraud be detected. After the definition of logs, we carried out the experiments with the algorithms described in Section 3. The following subsection describes the results.

4.2 Results

Table 1 reports a summary of results for three performance metrics, as follows: ACC indicates the accuracy of algorithm, that is, how many traces in the log were correctly classified; TPR, which is an acronym of true positive rate, indicates the ratio of anomalous (positive) traces that were correctly classified; and FPR, which is an acronym of false positive rate, indicates the ratio of normal (negative) traces that were incorrectly classified. These values were obtained for the algorithms presented in Section 3.3, which indicates the use of a threshold value equal to zero, defined based on the first quartile of our study with logs without anomalous traces (Section 3.2).

Table 1 Summary of assessment

	Fitness	*Size*	*Structural*	*Behavioural*
ACC	54.07%	54.07%	78.54%	78.61%
TPR	99.89%	99.89%	0.00%	1.67%
FPR	57.43%	57.43%	0.00%	0.36%

Notes: Threshold based on 1st quartile.

It can be noticed in Table 1 that structural and behavioural metrics are not appropriate to detect anomalous traces, while fitness and size metrics have had a TPR near to 100%. Also, despite similar semantic between size and structural metrics, which are metrics used to report the complexity of a model, they have had opposite performance. Such a differentiated behaviour may be related to an extended definition of size metric, which also considers the edges of Petri net.

However, despite the good performance of fitness and size metrics for classifying anomalous traces, their mismatch rates were bad for classify 'normal' traces. That behaviour in the fitness and size approaches induced us to believe that the threshold value needs to be greater than zero, in order to accommodate a greater compliance variance for 'normal' traces. For that reason, we redefined the algorithms based on fitness and size metrics, presented in Section 3.3, through the use of the following threshold values: *threshold* = 0.006782 for fitness metric; and *threshold* = 0.01594 for size metric. Such values represent the compliance variance mean in the study presented in Section 3.2.

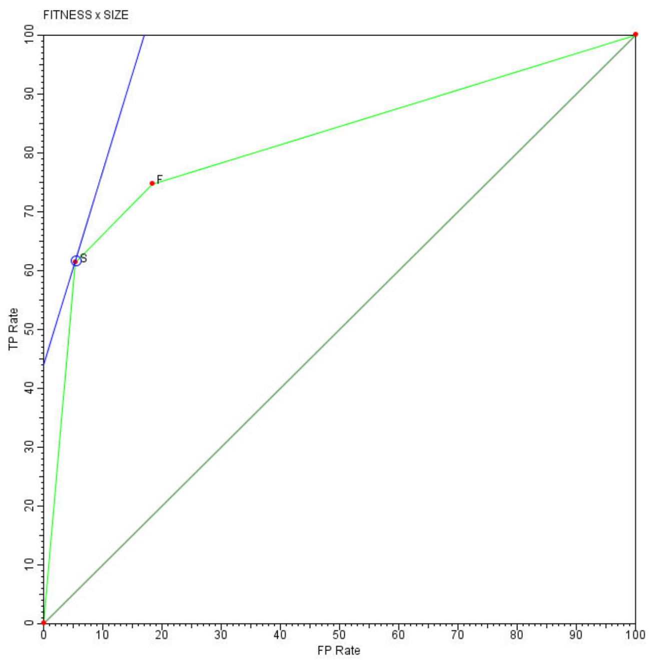
Then, we carried out tests with the redefined algorithms, whose results are reported in Table 2. Figure 4 depicts a graphical analysis tool [ROC curve (Fawcett, 2004)] utilised to compare the performance of the anomaly detection algorithms (or classifiers) based on fitness and size metrics. ROC curve is an useful technique for organising classifiers and visualising their performance. Each point in the figure represents one algorithm, labelled as follows: **F** for fitness approach; and **S** for size approach.

Table 2 Summary of assessment

	Fitness	*Size*
ACC	80.21%	90.92%
TPR	74.71%	61.38%
FPR	18.41%	5.37%

Note: Threshold based on mean.

Figure 4 ROC curve for fitness and size approaches (see online version for colours)



In an ROC curve, the best classifier is that one closer to optimal point of coordinates (0, 100), which means 0% of misclassification of anomalous traces and 100% of correct classification of anomalous traces. That is, the *x-axis* represents the ratio of false positives (FP rate), and the *y-axis* represents the ratio of true positives (TP rate). The diagonal line in the centre of graph is used to identify good or bad classifiers. Points above the diagonal line indicate good systems, while points below the line indicate bad systems. The figures illustrate a convex curve that is tangent to the iso-performance classifiers, that is, classifiers that are possibly optimal. Moreover, there is an upper line that is tangent to the curve, where it touches the optimal classifier. The inclination of that line depends on the number of positives examples (anomalous traces) in the data, which for this assessment were about 25% of anomalous instances in each log.

Among the algorithms, the fitness approach has had the best performance for detecting anomalous traces (TPR = 74.71%), but it also has had the worst performance for classifying 'normal' traces (FPR = 18.41%). Thus, for that reason, the size metric approach has had the best accuracy (ACC = 90.92%). Considering the execution time of algorithms, both fitness and size approaches have had similar performance. The average time of execution for algorithms was as follows: 3.42 s for fitness approach; 3.58 s for size approach. In the worst case, the algorithms were executed in the following times: 8.7 s for fitness approach; 9.5 s for size approach.

5 Conclusions and future work

Nowadays there is a huge demand for auditing system, especially motivated by financial accounting frauds. Scandals related to accounting and financial mismanagement of companies can reduce the confidence of investors, but the adherence of companies to best practices of governance, like COSO and COBIT, can support companies against frauds, attracting new investors. Even considering that the adoption of best practice of governance by companies influences the market confidence, such companies demand a flexible environment of business control for strategic reasons, otherwise their business processes can not be easily extended to new business models (e.g., by merging of two companies). Thus, there is clearly a demand for a balance between rigid control and a flexible and competitive control.

This paper showed a fraud detection algorithm for logs of PAS based on four different metrics: fitness, structural appropriateness, behavioural appropriateness, and size. We argued that such an algorithm is important in scenarios of application where a flexible and secure business process is essential. The anomaly detection algorithm was based on the α-algorithm, a process mining algorithm.

We analysed the algorithms with 435 logs synthetically created. Also, an ROC curve was generated for fitness and size approaches, which have had the best performance among the four alternatives. Then, the anomaly detection algorithm based on the size metric has shown the best accuracy, for it has correctly classified nearly 91% of traces from the logs. Regarding performance analysis, both fitness and size approaches have had similar execution time.

However, even considering the results of this work, some additional work has to be done. Among limitations, it is important to note that the accuracy of algorithms is strictly related to the following components:

1 the process mining algorithm

2 the metric used to evaluate the compliance variance between two logs (with and without anomalous traces)

3 the threshold value used to define the compliance variance limit for logs without anomalous traces.

Therefore, a future research agenda will consider the assessment of other process mining algorithms (e.g., extensions of α-algorithm), others metrics, and a deeper study of threshold values.

References

Agarwal, D.K. (2005) 'An empirical Bayes approach to detect anomalies in dynamic multidimensional arrays', in *ICDM*, pp.26–33.

Agrawal, R., Gunopulos, D. and Leymann, F. (1998) 'Mining process models from workflow logs', in *EDBT '98: Proceedings of the 6th International Conference on Extending Database Technology*, pp.469–483, Springer-Verlag, London, UK.

Bezerra, F. and Wainer, J. (2007a) 'Towards detecting fraudulent executions in business process aware systems', in *WfPM 2007 – Workshop on Workflows and Process Management*, in Conjunction with SYNASC 2007, Timisoara, Romania.

Bezerra, F. and Wainer, J. (2007b) 'Um método de detec cão de anomalias em logs de processos de negócios', in de Toledo, M.B.F. and Madeira, E.M. (Eds.): *I Brazilian Workshop on Business Process Management*, SBC, in Conjunction with Webmedia 2007, Gramado, RS, Brazil.

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

Bezerra, F. and Wainer, J. (2008b) 'Anomaly detection algorithms in logs of process aware systems', in *SAC '08: Proceedings of the 2008 ACM symposium on Applied Computing*, pp.951–952, ACM, New York, NY, USA.

Cook, J.E. and Wolf, A.L. (1998) 'Discovering models of software processes from event-based data', *ACM Trans. Softw. Eng. Methodol.*, Vol. 7, No. 3, pp.215–249.

Cook, J.E., Du, Z., Liu, C. and Wolf, A.L. (2004) 'Discovering models of behavior for concurrent workflows', *Computers in Industry*, Vol. 53, No. 3, pp.297–319.

de Medeiros, A.K.A. (2006) 'Genetic process mining', PhD thesis, Technische Universiteit Eindhoven, Eindhoven, ISBN 978-90-386-0785-6.

de Medeiros, A.K.A., van der Aalst, W.M.P. and Weijters, A. (2003) 'Workflow mining: current status and future directions', in Meersman, R., Tari, Z. and Schmidt, D. (Eds.): *On the Move to Meaningful Internet Systems, LNCS*, Vol. 2888.

de Medeiros, A.K.A., Weijters, A.J.M.M. and van der Aalst, W.M.P. (2006) 'Genetic process mining: a basic approach and its challenges', *Lecture Notes in Computer Science*, Vol. 3812, pp.203–215, ISSN 0302-9743.

Donoho, S. (2004) 'Early detection of insider trading in option markets', in *KDD '04: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.420–429, ACM Press, New York, NY, USA.

Fawcett, T. (2004) *Roc Graphs: Notes and Practical Considerations for Researchers*.

Fawcett, T. and Provost, F. (1997) 'Adaptive fraud detection', *Data Mining and Knowledge Discovery*, Vol. 1, pp.291–316.

Hammori, M., Herbst, J. and Kleiner, N. (2006) 'Interactive workflow mining – requirements, concepts and implementation', *Data Knowl. Eng.*, Vol. 56, No. 1, pp.41–63.

Herbst, J. and Karagiannis, D. (2004) 'Workflow mining with involve', *Computers in Industry*, Vol. 53, No. 3, pp.245–264.

Lee, W. and Xiang, D. (2001) 'Information-theoretic measures for anomaly detection', in *IEEE Symposium on Security and Privacy*.

Maruster, L., van der Aalst, W.M.P., Weijters, T., van den Bosch, A. and Daelemans, W. (2001) 'Automated discovery of workflow models from hospital data', in Krse, B., Rijke, M., Schreiber, G. and Someren, M. (Eds.); *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001)*, pp.183–190.

Noble, C.C. and Cook, D.J. (2003) 'Graph-based anomaly detection', in *KDD '03: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.631–636, ACM Press, New York, NY, USA.

Pandit, S., Chau, D.H., Wang, S. and Faloutsos, C. (2007) '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*, pp.201–210, ACM Press, New York, NY, USA.

Pinter, S.S. and Golani, M. (2004) 'Discovering workflow models from activities' lifespans', *Computers in Industry*, Vol. 53, No. 3, pp.283–296.

Rozinat, A. and van der Aalst, W.M.P. (2008) 'Conformance checking of processes based on monitoring real behavior', *Information Systems*, Vol. 33, No. 1, pp.64–95.

Rozinat, A. and van der Aalst, W.M.P. (2005) 'Conformance testing: measuring the fit and appropriateness of event logs and process models', in *Business Process Management Workshops*, pp.163–176.

Rozinat, A., de Medeiros, A.A., Günther, C., Weijters, A. and van der Aalst, W.M.P. (2007) 'Towards an evaluating framework for process mining algorithms', Technical report, Technische Universiteit Eindhoven, BETA Research School for Operations Management and Logistics.

Sabhnani, R., Neill, D. and Moore, A. (2005) 'Detecting anomalous patterns in pharmacy retail data', in *Proceedings of the KDD 2005 Workshop on Data Mining Methods for Anomaly Detection*.

Schimm, G. (2004) 'Mining exact models of concurrent workflows', *Comput. Ind.*, Vol. 53, No. 3, pp.265–281.

van der Aalst, W.M.P. and de Medeiros, A.K.A. (2005) 'Process mining and security: detecting anomalous process executions and checking process conformance', *Electr. Notes Theor. Comput. Sci.*, Vol. 121, pp.3–21.

van der Aalst, W.M.P. and Weijters, A.J.M.M. (2004) 'Process mining: a research agenda', *Computers in Industry*, Vol. 53.

van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G. and Weijters, A.J.M.M. (2003) 'Workflow mining: a survey of issues and approaches', *Data Knowl. Eng.*, Vol. 47, No. 2, pp.237–267.

van der Aalst, W.M.P., Weijters, T. and Maruster, L. (2004) 'Workflow mining: discovering process models from event logs', *IEEE Trans. Knowl. Data Eng.*, Vol. 16, No. 9, pp.1128–1142.

Wainer, J., Kim, K. and Ellis, C.A. (2005) 'A workflow mining method through model rewriting', in Fuks, H., Lukosch, S. and Salgado, A.C. (Eds.): *Groupware: Design, Implementation, and Use: 11th International Workshop*, CRIWG 2005, Vol. 3706, pp.184–191, Porto de Galinhas, Brazil.

Yang, W-S. and Hwang, S-Y. (2006) 'A process-mining framework for the detection of healthcare fraud and abuse', *Expert Systems with Applications*, Vol. 31, No. 1, pp.56–68.