Anomaly Detection using Process Mining

Fábio Bezerra[1], Jacques Wainer[1], and W.M.P. van der Aalst[2]

[1] Institute of Computing - UNICAMP
Av. Albert Einstein, 1251
Campinas, São Paulo, Brazil
{fbezerra,wainer}@ic.unicamp.br
[2] Dep. of Mathmatics and Computer Science - TU/e
Den Dolech 2, 5600 MB
Eindhoven, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Abstract. Recently, several large companies have been involved in financial scandals related to mismanagement, resulting in financial damages for their stockholders. In response, certifications and manuals for best practices of governance were developed, and in some cases, tougher federal laws were implemented (e.g. the Sarboness Oxley Act). Companies adhered to these changes adopting the best practices for corporate governance by deploying Process Aware Information Systems (PAISs) to automate their business processes. However, these companies demand a rapid response to strategic changes, so the adoption of normative PAISs may compromise their competitiveness. On one hand companies need flexible PAISs for competitiveness reasons. On the other hand flexibility may compromise security of system because users can execute tasks that could result into violation of financial loses. In order to re-balance this trade-off, we present in this work how ProM tools can support anomaly detection in logs of PAIS. Besides, we present the results of the application of our approach with a real case.

Key words: Process mining, anomaly detection, auditing systems

1 Introduction and motivation

Management trends in the early 1990's largely motivated the adoption of **P**rocess **A**ware **I**nformation **S**ystems (PAISs) by organizations [1]. The use of PAISs illustrates a shift from data to process-oriented systems, which clearly separates business process logic from application programs, facilitating redesign and extension of process models. Moreover, legal requirements are also motivating companies to adopt PAISs and follow best practices of governance (e.g. COBIT, Control Objectives for Information and related Technology) in order to support the control of their business processes. For example, we can cite the Sarbanes-Oxley Act, which is a United States federal law enacted in response to a number of major corporate and accounting scandals (e.g. Enron and WorldCom).

Despite the automation provided by PAIS, the business process control of competitive companies should not be supported by normative tools like a classi-

cal production WMS (Workflow Management System). These companies demand a flexible automation of their business processes, since they need to respond rapidly to new market strategies or new business models. On the other hand, a flexible system may be vulnerable to fraudulent or undesirable executions. These considerations illustrate the 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.

Therefore, there is clearly a demand for auditing systems, and buzzwords such as BAM (Business Activity Monitoring), BOM (Business Operations Management), and BPI (Business Process Intelligence) illustrate the interest of vendors to support the monitoring and analysis of business activities [2]. Besides, the spectacular growth of log data in the form of audit trails, transaction logs, and data warehouses, and the requirement from a BPM (Business Process Management) perspective, have stimulated and enabled the development of process mining techniques. The process mining is mainly concerned with the discovery of process models from logs generated by information systems [3, 4]. Recent developments in the field of process mining have led to a renewed interest in anomaly detection [5, 6, 7] and security issues [8]. Thus, this paper presents an approach to detect anomalous traces using available process mining tools of ProM framework[1].

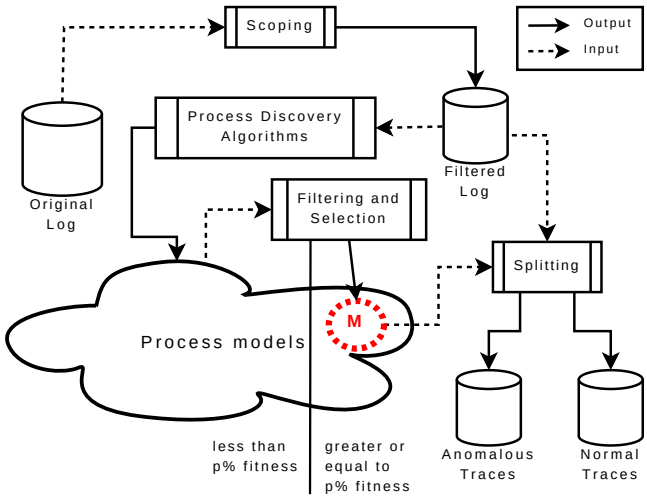

Fig. 1. Overview of our anomaly detection approach.

Figure 1 provides an overview of our proposed approach which is organized in five steps: (i) scoping, (ii) process discovery, (iii) filtering of fitting models, (iv) model selection, and (v) splitting of log. The *scoping* phase is a domain dependent step by applying some filters where instances and activities that are

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

out-of-scope are removed from the original log. The next two steps deal with *discovering* models and *filtering of fitting models*, i.e. the selection of models that satisfy a minimum (p%) fitness criteria - the degree of fitness refers to the ability to reproduce the log. Then, we *select the most appropriate* model among fitting models. An appropriate model is a structuraly simple and behavioraly specific model. Finally, we *classify the instances* of log in anomalous and normal instances using the selected model. In this approach, which focuses on analysis of control-flow perspective, if an execution trace in the log is not an instance of (or does not fit) the appropriate model, it is an anomalous trace.

The remainder of this paper is organized as follow. In Section 2 we present some related work in the area of process mining, conformance checking, trace clustering, and auditing. Albeit it is hard to present a precise definition for anomaly in process-aware context, specially when we consider very dynamic application domains (e.g. health care systems), in Section 3 we present what we believe to be a suitable anomaly definition. In Section 4 we present how ProM framework can be applied to operationalize this definition. Besides, we provide a case study in Section 5 to show how our anomaly detection approach can be applied in a real scenario, and we provide a final discussion and directions for future work in Section 6.

2 Related work

Process mining techniques allow for various types of analysis based on so-called event logs. For example, 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 [3, 9, 10]. The term was first coined in the context of software processes. Cook and Wolf, in [11], 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, in [12], present an algorithm that mine models having three properties in mind: completeness, minimality, and irredundancy.

Among the recent process mining approaches, the most visible one is the $\alpha-$algorithm [10, 4]. The effectiveness 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 severe limitations, for example, the inability to deal with short loops.

Noise in the event log is closely related to anomaly detection. Some process mining methods deal with the mining of noisy logs [12, 3, 13, 14, 15], 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 modeled in the resulting process model. A more sophisticated and promising approach, called genetic mining, was proposed in [16]. 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.

All previously mentioned process mining methods are mainly concerned with the modeling of normal behavior, yet some of them also deal with noisy logs. However, abnormal behavior was not deeply studied by process mining community, although it is a clearly important subject to the development of more accurate auditing systems. Then, in order to fill this gap, recent researches have been addressing the problem of identifying anomalous trace in logs of PAISs [8, 17, 7, 6]. In [8], 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, but a known "normal" log may not be available in applications domains that demand flexible support. In [17], 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. In [6] and [7], Bezerra and Wainer present three different approaches to detect anomalous traces: sampling, threshold, and iterative approaches. Nevertheless, as pointed out by the authors, the methods presented in [6, 7] have serious practical limitations, directly resulting from the adopted process mining algorithm, which can not deal with larger logs.

3 Formal anomaly definition

There are many meanings associated with the definition of anomaly. An anomaly can be an **exceptional** execution, a **noise** in the log, possibly caused by system failure or 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 lead to undesirable results from a business point of view. However, despite different meanings associated with the term anomaly, there are some common generic definitions such as: (i) a rare or infrequent event; (ii) a deviation from a normal form or rule; (iii) an unexpected result; or (iv) a state outside the usual range of variations.

Nevertheless, a precise definition of normal, norm, or rule is difficult, or even impossible, if one assumes a generic context, e.g. an arbitrary PAIS. Note that, in very dynamic environments, like health care systems, each instance (e.g. patient treatment) may be different from others, so each instance can be viewed as an unexpected occurrence. Next, we present a definition for anomalous traces. We believe that such a definition is a first step towards a more accurate and generic definition. We will make this definition operational using ProM framework, and we point out in Section 4 how ProM can address this definition.

Throughout this paper 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. Using the notion of a trace, we define the concept of an event 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 as the set of all traces defined over A and $T' \subseteq T$, then a log L is defined as $L \subseteq T'$[2]*.*

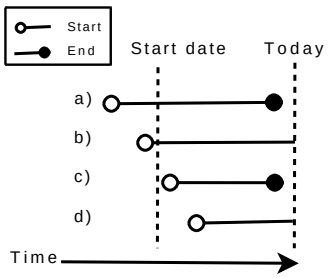

Fig. 2. Problems related with an imported log.

In the scoping step of our anomaly detection approach (see Figure 1) the domain analyst will define which activities and traces may be removed from log before anomaly detection. We call the first step scoping because it represents the moment when the domain analyst defines what is important to consider in the analyses. Also, traces that are clearly not fully recorded should be removed. For example, we show in Figure 2 four traces (a, b, c, and d) from a log, and we indicate with dashed lines the period that was used to import the traces for analysis. Thus, it is clear in this figure that: (i) trace a) should be removed because it does not have the expected start activity; (ii) trace b) should be removed because it does not have the expected start and end activities; and (iii) trace d) should be removed because it does not have the expected end activity. The scoping step is formally defined below.

Definition 3 *Scoped Log.*
Given a log L as defined in Definition 2, and a set A^S of scoped activities such

[2] Note that for simplicity we assume that a log is a set of traces. However, in reality a log is a bag (i.e. multiset) of traces since each sequence of activities may appear multiple times in the log. Although we use set in our formal definition, our implementation in ProM takes frequencies of traces into account.

that $A^S \subseteq A$. Then, an scoped log L^S is a set of traces t based on scoped activities A^S such that:

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

where filter removes all activities in t that are not in A^S, and complete(t) is a boolean function that evaluates to false if t is not complete or inappropriate.

In order to classify the traces of a log as anomalous and normal, we have to use what we call an *appropriate model*, which is a model that has a minimum fitness support (see Definition 5) and maximizes a function called appropriateness (see Definition 6). The minimum fitness support is a parameter used to filter the models that can be discovered from the log, that is, among the models (possibly infinitely many) we are interested in the models that can classify at least $p\%$ of traces as normal, where $p\%$ refers to the minimum fitness support.

Definition 4 *Fitness Instance Test Function.*
$f_M : L \rightarrow \mathbb{B}$ is the fitness instance test function that indicates if a trace from a log L is an instance of a model M. A trace t is instance of a model M if t can be completely parsed by M. It can be defined as follows:

$$f_M(t) = \begin{cases} true, & \text{if } t \text{ can be replayed by model } M \\ false, & \text{otherwise} \end{cases}$$

Definition 5 *Fitness Model Test Function.*
It is a function $f : \{(M, L) | M \text{ is a model} \wedge L \text{ is a log}\} \rightarrow [0, 1]$ that indicates the degree of fitness between a model M and a log L, that is, how many traces from log L fit or can be completely parsed in model M. Function f is defined as follows:

$$f(M, L) = \frac{|\{t \in L | f_M(t)\}|}{|L|}$$

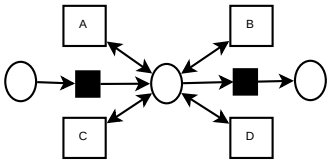

Fig. 3. Example of a generic model (Flowered model).

Therefore, the *fitness model test function* indicates how much of the observed behavior in the log can be supported by a model. That is, a fitness of 100% means that the model supports the whole log, so it is able to replay each trace from the log correctly. Nevertheless, a model with 100% of fitness does not mean an appropriate model. For example, the generic model depicted in Figure 3 can replay whatever trace defined over the set of activities {A, B, C, D}, so this model will never be able to detect anomalous traces in a log whose traces are

based on these activities. On the other hand, a model with low fitness value would classify many traces in log as anomalous. Hence, *appropriateness test function* is important to help us choose which fitting model is more appropriate, that is, given two fitting models which one better describes the log in a simple and specific way. Therefore, we present a formal definition of *appropriateness test function*, which supports the fourth step of our anomaly detection approach, the model selection step. Then, after selecting the *appropriate model*, a trace from the log is anomalous if it is not fitting model (cf. Definition 7).

Definition 6 *Appropriateness Test Function.*
$a : \{(M, L) | M \text{ is a model} \wedge L \text{ is a log}\} \rightarrow [0, 1]$ *is a function that indicates how appropriate is a model* M *when compared with log* L*, where appropriate means that a simple model is preferable than complex one, and that "too much" additional behavior is undesirable. Therefore, such a function represents a balance between structural complexity and extra-behavior support.*

Finally, once we selected an appropriate model, we perform the last step of our anomaly detection approach, the splitting of log in two sets: *anomalous traces* and *normal traces*. Below, we present a formal definition of anomalous trace.

Definition 7 *Anomalous Trace.*
Given log L, $p \in [0, 1]$ *the desired minimal degree of fitness between a model and a log, and* M^* *an* appropriate model *such that:*

- $f(M^*, L) \geq p$;
- $\forall M' \ f(M', L) \geq p \Rightarrow a(M', L) \leq a(M^*, L)$.

Then, an anomalous trace $t' \in L$ *is defined as follows:* $\neg f_{M^*}(t)$*, i.e.* $\{t \in L \mid \neg f_{M^*}(t)\}$ *is the set of anomalous traces.*

Summarizing, among the models that can be discovered from a scoped log L^S, we are interested in the model M^*, which we call *appropriate model* and has a minimum fitness degree p, but whose appropriateness is greater or equal to the appropriateness of all others models with minimum fitness p that can also be discovered from this log L^S. Then, the **anomalous traces** are those traces from log that do not fit the *appropriate model* M^*. In the following section we address this formal anomaly definition operational by using ProM.

4 Application based on ProM

The ProM framework is a pluggable environment for process mining [18]. It is platform independent as it is implemented in Java, and it is open-source. The framework is flexible with respect to the input and output format, and it is also open enough to allow for the easy reuse of code during the implementation of new process mining techniques. ProM supports the analysis of three main perspectives: (i) the process perspective that focuses on the control-flow mining; (ii) the

organizational perspective that focuses on the performers of activities; and (iii) the case perspective that focuses on properties, data, and values manipulated by activities. Because our anomaly detection approach is focusing on control-flow deviations, we are specially interested in the plug-ins dealing with process perspective in ProM. In this section, we show how the ProM framework can be used in the identification of anomalous traces based on our formal definition (cf. Definition 7).

4.1 Scoping

The first step of our anomaly detection approach is concerned with the removal of activities and traces from log that are not interesting for analysis or that may lead the definition of anomalies that are the result of an incomplete log. ProM has a lot of log filtering tools that can be applied in this step. For example, in ProM is possible to indicate what are the start and end activities of traces from log, so every trace that does not start and end with selected activities will be removed from log.

ProM also provides inspecting tools that can be used to evaluate the frequency of activities. Using filtering it is possible to perform an analysis based only on frequent traces. Besides, ProM provides an analysis plug-in called *LTL Checker* that can be used to filter traces that satisfy certain properties, for example, traces with a causal relation between two activities.

4.2 Process discovery and filtering

The next two steps of our anomaly detection approach address the discovery and filtering of models. The process discovery step deals with the automated construction of a process model that describes the log used during discovery, while the filtering step is related with the selection of models that satisfy a minimum fitness constraint (the *p value* in Definition 7). In order to address the discovery process step, ProM provides several algorithms, and all available process discovery algorithms can be used. On the other hand, the fitness instance test function, as described in Definition 4, is not provided separately by ProM, yet it can be obtained indirectly through the *conformance checker* plug-in [18]. The *fitness*(f) metric of conformance checker plug-in is a more fine-grained metric that evaluates how much a model fits a log considering both trace and activity perspectives.

Moreover, the fitness of a model can be evaluated through a metric in ProM called *PM* (*Parsing Measure*) that directly supports Definition 5. Such a metric can be used with *control-flow benchmark* plug-in, but it works only with heuristic models, and because there is not a direct conversion plug-in from Petri nets to heuristic models, we can not use this metric with process mining algorithms that output Petri nets models. On the other hand, we can accomplish this limitation using conformance checker plug-in, which provides an interface where it is possible to select only the fitting traces (100% of fitness), and then we can see the percentage of traces that fits the model.

4.3 Model selection

Model selection is the fourth step of our approach, and it is concerned with the selection of what we call *appropriate model*, that is, a simple and non-generic model. In order to objectively help us choose such an appropriate model we need an appropriateness test function that supports Definition 6. Although ProM does not provide a plug-in that directly selects the most appropriate model, the appropriateness metrics implemented in both *conformance checker* and *control-flow benchmark* plug-ins can be used in for a suitable definition of an appropriateness test function (cf. Equation (1)). Hence the appropriateness test function may be evaluated in ProM as follows:

- using a metric called *structural appropriateness*, which assesses the complexity of a model, and we represent here as a function $f_S(M)$, where M is a model;
- using a metric called *behavioral appropriateness*, which assesses how specific is a model regarding a log, and we represent here as a function $f_B(M, L)$, where M is a model and L is a log;
- finally, since both functions are defined for the same codomain ($[0, 1]$), we could objectively define appropriateness as a balance value between these structural and behavioral metrics, as follows:

$$a(M, L) = \frac{f_S(M) + f_B(M, L)}{2} \tag{1}$$

4.4 Splitting

Finally, since we have an *appropriate model*, the last step of our anomaly detection approach can be easily achieved through *conformance checker* plug-in of ProM. That is, once we have got a model that supports a minimum fitness threshold (value p of definition), and such a model also has the greatest appropriateness value amongst other models, we can simply select those traces that do not fit the model as follows: (i) selecting fitting traces as normal traces; and then (ii) inverting selection to identify the anomalous traces.

5 Municipal household support system

In this section we present a real application of ProM tools for supporting our anomaly detection approach. It refers to a log of the information system of the Dutch municipality. The process is about supporting citizens that need help in the form of a wheelchair, scootmobiel, adaptation of house (elevator), and household help. The log used in this analysis comprises event data from January 2007 to August 2008, and it contains information of 876 process instances that together represent 5497 activities, among 10 different activities available in the log. Besides, the shortest trace from log has 1 activity, while the longest has 12 activities. On average, the traces have 6 activities.

Because many models can be discovered from a log (maybe infinite), and considering the lack of automated tools to generate all possible candidates, we explored the set of possible process models in a semi-automatic fashion, i.e., the appropriate model was discovered through manual parameter selection. In the following we present how we applied our anomaly detection approach.

5.1 Scoping

During scoping, we first made an analysis based on frequencies of start and end activities. As stated in Section 3, depending of period used to import the log, some traces may start and/or end with an intermediate activity. These incomplete traces were removed. Then we applied the following filters on the original log, which were also supported by users of the system.

– define "Request registration" as the unique start activity because it is a predominant start activity, as we can notice in Table 1;
– define "Final Phase" as the unique end activity (see Table 1);

Table 1. Frequency of start and end activities obtained from ProM

Frequency of start activities		Frequency of end activities	
Activity	Frequency	Activity	Frequency
Request registration	96,12%	Final Phase	94,52%
Reporting & Decision	3,43%	Reporting & Decision	2,06%
Private research	0,34%	Request registration	1,03%
Research	0,11%	Left filing	0,91%
		Keys and decide	0,69%
		Accounting	0,34%
		Waiting recovery	0,23%
		Research	0,11%
		Return	0,11%

In the end of scoping step we obtained a log with 796 traces that as a whole comprise 5191 activities. Besides, the shortest trace from log has 5 activities, while the longest has 12 activities. On average, the traces have 6 activities.

5.2 Discovering, filtering, and selection

Our proposal approach deals with the search of an appropriate model, which satisfies a minimum fitness and maximizes appropriateness. Figure 4 depicts three models that we mined from the scoped log, and their respective properties (**f** for fitness, **s** for structural appropriateness, **b** for behavioral appropriateness, and **a** for appropriateness). We considered 80% as the minimum fitness support in this analisys. We used heuristics mining plug-in for process discovery because it is robust for noise and exceptions since it outputs a model based on frequent patterns.

Then, we got the Petri net A (after converting from a heuristic net model). Specifically in the case of this log, whose activity frequencies are reported in

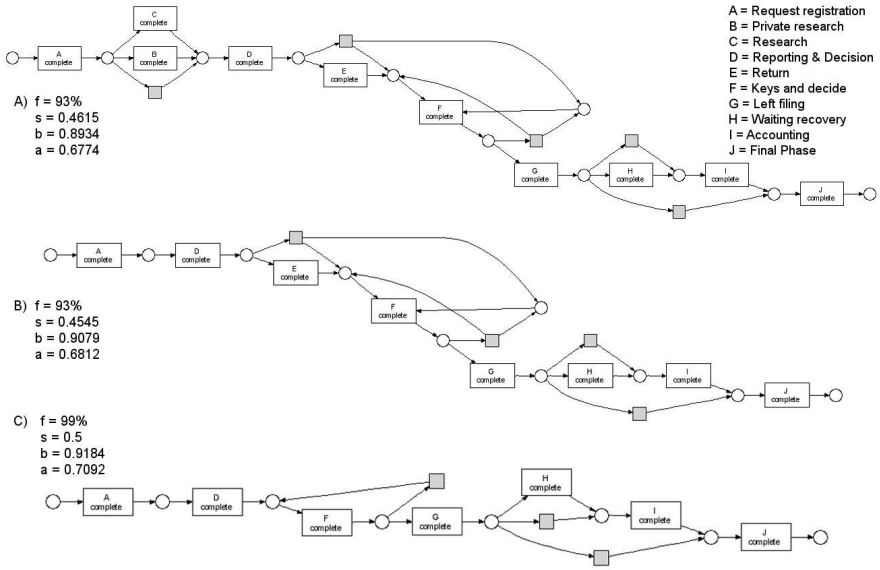

Fig. 4. Petri net models based on frequency filtering analysis.

Table 2. Activity Frequencies

Model element	Occurrences (relative)
Keys and decide	16,41%
Reporting & Decision	15,43%
Left filing	15,39%
Request registration	15,33%
Final Phase	15,33%
Accounting	11,42%
Waiting recovery	9,59%
Return	1,00%
Private research	0,04%
Research	0,04%

Table 2, the two most infrequent activities ("Private research" and "Research") add an unnecessary complexity to model A although they are significantly infrequent when compared with other activities. For that reason, we applied heuristic mining over a filtered version of scoped log, which does not consider activities "Private research" and "Research". This way, we got Petri net B, which is a model more appropriate than model A.

However, although "Return" activity is significantly more frequent than "Private research" and "Research" activities, it is also significantly infrequent when compared with other activities of log (see Table 2). That is, "Return" activity adds an unnecessary complexity to model in Figure 4 A. For that reason, we also mined scoped log, but filtered from "Private research", "Research", and "Return". As a result, we obtained the Petri net C, which is more appropriate than other models, and it also has a better fitness. Therefore, we selected Petri net C as the appropriate model, so it was utilized for splitting step. Note that

the selection of this model was not automated and we did not do an exhaustive search. Moreover, manual inspection showed that this is indeed the most appropriate model having a fitness of at least 80%.

5.3 Splitting

Finally, we got the fitting and non-fitting traces using the appropriate model (Petri net C in Figure 4). In this analysis we considered 80% for **p** (minimum fitness support), so we supported to find at most 20% of anomalous traces in the log. However, because we got an appropriate model whose fitness was 99%, we detected only 6 anomalous traces from a total of 796 traces of *scoped log*.

6 Conclusion and future work

Recent management trends and the adoption of rigorous best practices of corporate governance stimulated companies to deploy PAIS in order to automate and control their business processes, and also to track misuse of their systems (e.g. financial scandals related to mismanagement). However, the control provided by normative systems may compromise the necessary flexibility to companies in being agile and competitive in the market. This work presents an approach to identify anomalous traces, which may represent a misuse, for deal with this problem. For example, the identification of anomalous traces can lead to an investigation and probable evolution of the business process models. Our approach is based on a formal definition of anomalous trace, which is defined through two parameters: (i) fitness model degree (p%); and (ii) appropriateness of model (a). We described how ProM framework can be utilized for support this formal definition. Then, we carried out an application of approach with a real log from a Dutch municipality.

The presented anomaly detection approach is limited to the control-flow perspective. For example, fraud may follow a normal flow, but producing anomalous data (e.g. very large amount of money) or being executed by unauthorized roles or users (e.g. violation of four eyes principle). Therefore, we believe that data and organizational perspectives should also be considered to provide more accuracy, yet they may require a more complex anomaly detection framework. Because our approach relies on the selection of an *appropriate model*, we believe that a precise appropriateness metric should be defined. Besides, we think that an automated solution might be implemented, for example, through the use of genetic algorithms.

References

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

2. Rozinat, A., van der Aalst, W.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**(1) (March 2008) 64–95
3. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: A survey of issues and approaches. Data & Knowledge Engineering **47**(2) (November 2003) 237–267
4. van der Aalst, W.M.P., Weijters, A.J.M.M.: Process mining: a research agenda. Computers in Industry **53**(3) (April 2004) 231–244
5. Bezerra, F., Wainer, J.: Towards detecting fraudulent executions in business process aware systems. In: WfPM 2007 - Workshop on Workflows and Process Management, Timisoara, Romania (September 2007) In conjunction with SYNASC 2007.
6. Bezerra, F., Wainer, J.: Anomaly detection algorithms in logs of process aware systems. In: SAC '08: Proceedings of the 2008 ACM symposium on Applied computing, New York, NY, USA, ACM (2008) 951–952
7. Bezerra, F., Wainer, J.: Anomaly detection algorithms in business process logs. In: ICEIS 2008: Proceedings of the Tenth International Conference on Enterprise Information Systems. Volume AIDSS., Barcelona, Spain (June 2008) 11–18
8. van der Aalst, W.M.P., 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) (February 2005) 3–21
9. de Medeiros, A.K.A., van der Aalst, W.M.P., Weijters, A.: Workflow mining: Current status and future directions. In Meersman, R., Tari, Z., Schmidt, D., eds.: On The Move to Meaningful Internet Systems. Volume 2888 of LNCS. (2003)
10. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering **16**(9) (September 2004) 1128–1142
11. Cook, J.E., Wolf, A.L.: Discovering models of software processes from event-based data. ACM Trans. Softw. Eng. Methodol. **Vol. 7**(3) (1998) p. 215–249
12. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: EDBT '98: Proceedings of the 6th International Conference on Extending Database Technology, London, UK, Springer-Verlag (1998) 469–483
13. Cook, J.E., Du, Z., Liu, C., Wolf, A.L.: Discovering models of behavior for concurrent workflows. Computers in Industry **53**(3) (2004) 297–319
14. Pinter, S.S., Golani, M.: Discovering workflow models from activities' lifespans. Computers in Industry **53**(3) (2004) 283–296
15. Herbst, J., Karagiannis, D.: Workflow mining with inwolve. Computers in Industry **53**(3) (2004) 245–264
16. de Medeiros, A.K.A., Weijters, A.J.M.M., van der Aalst, W.M.P.: Genetic process mining: A basic approach and its challenges. In: Business Process Management Workshops. Volume 3812 of Lecture Notes in Computer Science., Nancy, France (September 2006) 203–215 ISBN 978-3-540-32595-6.
17. Yang, W.S., Hwang, S.Y.: A process-mining framework for the detection of healthcare fraud and abuse. Expert Systems with Applications **31**(1) (July 2006) 56–68
18. van Dongen, B., de Medeiros, A., Verbeek, H., Weijters, A., van der Aalst, W.: The prom framework: A new era in process mining tool support. In: Applications and Theory of Petri Nets 2005. Volume 3536 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2005) 444–454