Recommendation of Process Discovery Algorithms Through Event Log Classification

Damián Pérez-Alfonso[1(✉)], Osiel Fundora-Ramírez[2], Manuel S. Lazo-Cortés[3], and Raciel Roche-Escobar[1]

[1] University of Informatics Sciences, Habana, Cuba
{dalfonso,roche}@uci.cu
[2] XETID, Habana, Cuba
ofundora@xetid.cu
[3] National Institute for Astrophysics Optics and Electronics, Puebla, México
mlazo@inaoep.mx

Abstract. Process mining is concerned with the extraction of knowledge about business processes from information system logs. Process discovery algorithms are process mining techniques focused on discovering process models starting from event logs. The applicability and effectiveness of process discovery algorithms rely on features of event logs and process characteristics. Selecting a suitable algorithm for an event log is a tough task due to the variety of variables involved in this process. The traditional approaches use empirical assessment in order to recommend a suitable discovery algorithm. This is a time consuming and computationally expensive approach. The present paper evaluates the usefulness of an approach based on classification to recommend discovery algorithms. A knowledge base was constructed, based on features of event logs and process characteristics, in order to train the classifiers. Experimental results obtained with the classifiers evidence the usefulness of the proposal for recommendation of discovery algorithms.

Keywords: Process discovery · Process mining · Classification

1 Introduction

The execution of business processes on information systems is recorded on event logs. Process mining involves discovery, conformance and enhancement of process starting from event logs. A process discovery algorithm is a function that maps an event log onto a process model, such that the model is "representative" for the behavior seen in the event log [1]. Several algorithms have been developed for discovering process models that reflect the actual execution of processes. These models allow business analysts to make performance evaluations, anomaly identification, compliance checking, among others analysis.

Noise, duplicate tasks, hidden tasks, non-free choice constructs and loops are typical problems for discovery algorithms [2]. These problems are related to unstructured processes, commonly present in real environments [3]. Thus,

J.A. Carrasco-Ochoa et al. (Eds.): MCPR 2015, LNCS 9116, pp. 3–12, 2015.
DOI: 10.1007/978-3-319-19264-2_1

algorithms performance depends on event log characteristics and their associated process.

Several executions of different discovery algorithms could be required trying to obtain a quality model, thus becoming a time consuming and error-prone task. Selecting the right algorithms is a hard task due to the variety of variables involved. Several techniques execute different discovery algorithms for an event log and evaluate their resulting models using quality metrics [4]. Nevertheless, this empirical evaluation approach is computationally expensive and time consuming.

On the other hand, a recommendation technique is based on regression, but it requires reference models [5]. Reference models are not commonly available in contexts where process discovery is required. If there are reference models, it is unwise to assume that they reflect the actual execution of processes.

Studies that attempt to establish the algorithms with better performance under certain conditions have been published using the aforementioned empirical evaluation techniques [6]. However, the impact of each condition on model quality is not clearly defined yet. Therefore, the actual use of these studies remains limited.

The aim of this paper is to evaluate the usefulness of an approach based on classification to recommend discovery algorithms [7]. A knowledge base is constructed considering event log features such as: control-flow patterns, invisible tasks and infrequent behavior (noise). The recommendation procedure based on classification is tested over the knowledge base with different classifiers.

The paper is structured as follows: in the next section concepts and approaches related to process discovery and recommendation of discovery algorithms are presented. In Sect. 3, phases of the recommendation procedure, followed by a description of the creation process of the knowledge base are presented. In Sect. 4, experimental results of the classification based recommendation and their respective analysis are provided. A set of current techniques and approaches to assess and recommend discovery algorithms are discussed in Sect. 5. Finally, the last section is devoted to conclusions and outlines for future work.

2 Recommendation of Process Discovery Algorithms

A process discovery algorithm constructs a process model starting from an event log. An event is the occurrence of an activity of a process and a trace is a non-empty finite sequence of events recorded during one execution of such process. So, an event log is a multi-set of traces belonging to different executions of the same process.

Obtaining a quality model is the main goal of a process discovery algorithm. There are various metrics and approaches for estimating process model quality, though there is a consensus on the following quality criteria [1]:

- *Fitness:* The model should allow the behavior present in the event log.
- *Precision:* The model should not allow a behavior unrelated to the one stored in the log.

– *Generalization:* The model should generalize the behavior present in the log.
– *Simplicity:* The model should be as simple as possible. Also referred as structure, is influenced by the vocabulary of modeling language.

These criterion compete among them due to the inverse relationship between generalization and precision. A too general model could lead to allow much more behavior than the one presents in the log, it is also known as underfitting model. On the contrary, a too precise or overfitting model is undesirable. The right balance between overfitting and underfitting is called *behavioral appropriateness*. The *structural appropriateness* of a model refers to its ability to clearly reflect the performance recorded with the minimal possible structure [8]. A quality model requires both, behavioral appropriateness and structural appropriateness [9].

In order to obtain a quality model, a discovery algorithm should tackle several challenges related to event logs characteristics. Heterogeneity of data sources from real environments, can lead to difficult cases for discovery algorithms [10]. Infrequent traces and data recorded incompletely and/or incorrectly can induce wrong interpretations of process behavior. Moreover, data provided by parallel branches and ad-hoc changed instances generate complex sequences on event logs, this creates traces that are harder to mine.

Process structure is another source of challenges for discovery algorithms. Presence of control-flow patterns like non-free choices, loops and parallelism affect the discovery algorithms. For example, algorithms such as α, α^{+}, $\alpha^{\#}$ and α^{*} do not support non-free choices [11]. On the other hand, DWS Mining and α^{++} can deal with non-free choice but cannot support loops [2].

In order to identify which discovery algorithm allows obtaining suitable models for particular situations, a set of techniques for algorithms evaluation have been developed. Performance of these algorithms is determined through evaluation of quality of obtained models. Defined quality metrics are grouped under two main methods [12]. One method compares the discovered model with respect to the event log and is called *model-log*. The other method, called *model-model*, assesses similarity between discovered model and a reference model of process.

Evaluation frameworks allow end users to compare the performance of discovery algorithms through empirical evaluation with quality metrics [4,13]. Moreover, recommending a discovery algorithm for a given event log, based on empirical evaluation, involves time and resource consumption for each of the algorithms chosen as a possible solution. So, alternative approaches, based on classification, have been proposed to recommend process discovery algorithms [5,7].

3 Classification of Event Logs for Recommendation of Process Discovery Algorithms

In this section we evaluate the usefulness of an approach based on classification to recommend discovery algorithms [7]. *Classification* is the problem concerning the construction of a procedure that will be applied to a continuing sequence of cases, in which each *new case* must be assigned to one of a set of *pre-defined*

classes on the basis of *observed attributes or features* [14]. An event log on which is necessary to recommend a discovery algorithm is considered as a *new case* to be classified. The recommended algorithm is the *pre-defined class* to be assigned to an event log based on its *observed features*.

Taking into account the challenges for discovery algorithms the classification mechanism for recommendation of discovery algorithms consider the following factors:

1. The event log is the main information source that is available in all environments for process characterization.
2. The peculiarities of event log. must be considered in addition to process characteristics.
3. The results obtained by quality metrics on discovered models provide information about performance of process discovery algorithm facing event logs and process characteristics.

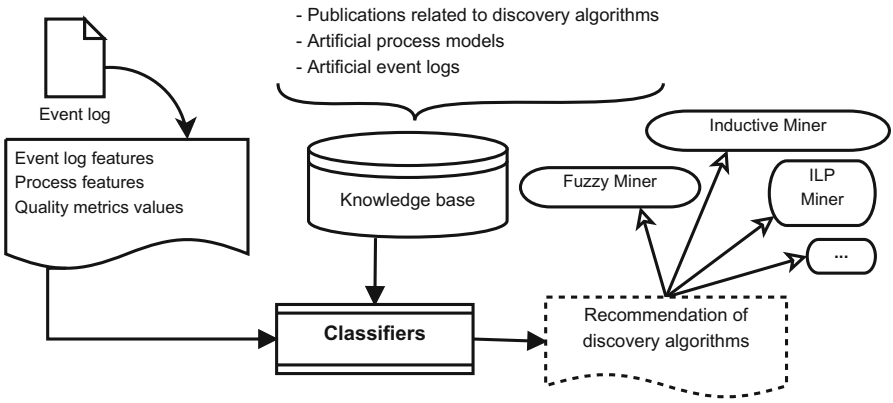

Fig. 1. Recommendation of discovery algorithms through event log classification

The stages for classification of event logs in order to recommend process discovery algorithms can be observed in Fig. 1. It shows that staring point is the new case to be classified. This new case is composed of event log and the process features that affect discovery algorithms and desired values for quality metrics on each quality criterion. The discovery algorithm that could discover a model for that log with the desired values on quality metrics is the *class*. The classifiers are trained in a knowledge base composed by cases with the same structure of aforementioned case, but labeled with the corresponding discovery algorithm. The discovery algorithm selected as the class for the new case is recommended to be applied on the new event log in order to obtain a process model with the specified quality values.

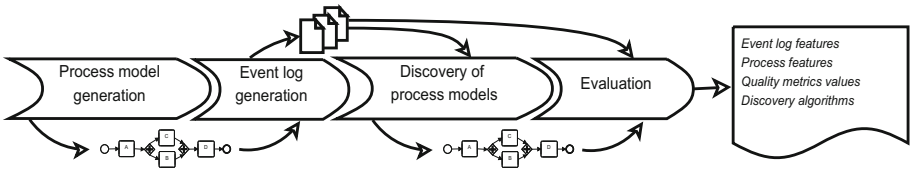

Fig. 2. Generation of artificial cases for the knowledge base

3.1 Building the Knowledge Base

A major challenge for this classification problem is the building of a knowledge base. In order to construct the knowledge base for this problem the following features were selected to build the cases:

1. The event log features.
2. The process characteristics.
3. The discovery algorithm used to obtain the process model.
4. The quality metrics obtained values based on the discovered model.

The sequence of phases followed to generate the artificial cases for the knowledge base is presented in Fig. 2. The outcome of each phase is used as input to the following.

The goal of the first and second phases is to obtain the event log and process features that affect discovery algorithms. In these two phases was used the *Process log generator* tool [15]. Using this tool several process models were generated in a random way. 67 of these process models that combine loops, non-free choice and invisible tasks were selected for the second phase. These process features were considered due to their impact on process discovery algorithms [6].

In order to generate the event logs a factorial complete experimental design was performed. For this factorial design five classes were considered: noise (C_1), noise interval (C_5), loops (C_3), parallelism (C_4) and invisible tasks (C_5). Noise could appear on different proportion of traces on event logs, on this cases 5 different proportions were used: 0, 25, 50, 75, 100. The same stand for noise interval: 0, 25, 50, 75 and 100 were the distribution used. Control flow patterns (C_3, C_4) and invisible tasks (C_5) were considered on a boolean manner. Formula 1 was used to calculate the required number of event logs to combine the aforementioned criteria.

$$\begin{aligned} F &= C_1 * C_2 * C_3 * C_4 * C_5 \\ F &= 5 * 5 * 2 * 2 * 2 = 200 \end{aligned} \tag{1}$$

Considering results from Formula 1, 201 event logs were generated combining the five features already stated. One third of these event logs has 500 traces each one. The second third has 1000 traces on each event log, while the last third has 1500 traces on each one.

On the third phase, discovery algorithms are applied on the generated event logs. Even there are several discovery algorithms, only five where selected for

this knowledge base. First criterion used on the selection was that the algorithm could discover a model on Petri net notation, or another notation that could be translated to a Petri net. This requirement is related to available quality metrics, that could be applied only on Petri net models. Performance of discovery algorithms on real and artificial event logs are considered too.

Therefore, published results about assessment of discovery algorithms [2,4,6,11] lead us to select the Heuristic Miner [16], ILP Miner [17], Inductive Miner [18], Genetic Miner [19] and Alpha Miner [20]. All these algorithms are available as plug-ins on the process mining framework ProM [21]. So, ProM 6.3 was used to obtain five process model from each event log using every selected algorithm.

The main goal of the last phase in the generation of artificial cases for the knowledge base is to evaluate the performance of discovery algorithms. One quality metric was selected for each quality dimension or criterion. So, were selected *fitness* [22], ETC [23], ARC Average [22] and Behavioral Generalization [24]. All these metrics belong to *model-log* method and are implemented in CoBeFra [25], a benchmarking tool. Therefore, using the generated event logs and the discovered models, the values for these quality metrics were obtained using CoBeFra.

Once all the phases were executed the information obtained were used to create the cases. One case was created for each discovered model. Each case is represented as a vector $c_i = \{at_i, aa_i, and_i, xor_i, l_i, it_i, nd_i, ni_i, f_i, p_i, g_i, s_i, DA\}$. In this vector i refers to the ordinal number of the discovered model. The at_i and aa_i variables stand for amount of traces and amount of activities in the event log used to discover the i model. Moreover, and_i, xor_i, l_i and it_i refers to the amount of parallelism, exclusive choice, loops and invisible tasks respectively, in the process related to the i model. The noise distribution and interval on the event log used to discover the i model are represented as nd_i and ni_i. Variables f_i, p_i, g_i and s_i express the values of the quality metrics obtained on the i model, related to fitness, precision, generalization and simplicity, respectively. Last but not least, DA is the discovery algorithm used to create the i model and this is the class that labeled the case.

Following the aforementioned description, a knowledge base was constructed with 795 cases. A ProM plug-in was developed to visualize and manage the knowledge base. This plug-in allows integration with other techniques in ProM.

4 Testing Classifiers

In order to find suitable classifiers for the knowledge base built, a set of well-known classifiers were trained and assessed. Before training, the data set was normalized to values between 0 and 1. Results for each classifier training are presented in Table 1, expressed in terms of Incorrectly Classified Instances (ICI) and Mean Absolute Error (MAE). Classifiers implementation on WEKA [26] where used in all cases, with default configuration values.

Based on the results presented on Table 1, seven classifiers were selected: Classification Via Regression, Multilayer Perceptron, Simple Logistic, Logistic, J48, Filtered Classifier and MultiClass Classifier. A ProM plug-in was developed

Table 1. Results of classifiers training

Classifiers	Cross-validation		Use training set		Percentage split	
	ICC	MAE	ICC	MAE	ICC	MAE
Classification via regression	231	0.2956	175	0.2644	80	0.3042
Multilayer perceptron	259	0.3329	196	0.2822	89	0.332
Simple logistic	264	0.3547	257	0.3421	87	0.3529
Logistic	286	0.3411	266	0.3314	103	0.348
PART	309	0.357	216	0.3282	136	0.3761
J48	317	0.3475	273	0.3278	103	0.359
OneR	325	0.4521	289	0.4263	103	0.4389
Bayes net	334	0.3724	334	0.3671	97	0.3454
Filtered classifier	343	0.3631	201	0.3508	135	0.3612
MultiClass classifier	343	0.3797	334	0.3752	126	0.3813
IBK	452	0.5317	280	0.4186	158	0.5389
Naive bayes	452	0.4234	474	0.4135	168	0.4254
K*	495	0.4329	525	0.4329	208	0.4334

Table 2. Execution times for empirical evaluation and classification of new event logs

Event log	Case vector	Empirical evaluation	Classification
1	$c_1 = \{500, 26, 2, 4, 1, 0, 0, 50, 1, 1, 1, 1, ?\}$	12 h 22 m 43 s	10 s
2	$c_2 = \{1000, 17, 1, 2, 1, 2, 25, 75, 1, 1, 1, 1, ?\}$	1 h 30 m 8 s	9 s
3	$c_3 = \{1500, 18, 2, 1, 2, 2, 0, 75, 1, 1, 1, 1, ?\}$	25 h 26 m 53 s	15 s
4	$c_4 = \{500, 25, 4, 1, 0, 0, 100, 25, 1, 1, 1, 1, ?\}$	37 h 10 m 23 s	40 s
5	$c_5 = \{500, 25, 6, 9, 2, 2, 25, 25, 1, 1, 1, 1, ?\}$	14 h 24 m 48 s	10 s

to integrate the WEKA implementation of these classifiers into ProM. Using this classification plug-in, the classifiers could be trained in ProM with the previously mentioned knowledge base. With the trained classifiers, the plug-in enables the recommendation of discovery algorithms through classification of new cases.

Five new event logs were generated to assess the recommendation provided by the classification plug-in developed. Empirical evaluation of discovery algorithms on these event logs were used as reference for this assessment. Starting from the features of these event logs, five new cases were prepared (Table 2). Each case has the structure $c_i = \{at_i, aa_i, and_i, xor_i, l_i, it_i, nd_i, ni_i, f_i, p_i, g_i, s_i, DA\}$.

Recommendation through classification means a significant time improvement with respect to empirical evaluation, as can be seen in Table 2. Besides, for each event log (with the exception of event log 4) the class proposed by six of the seven classifiers match with the discovery algorithm with best results on empirical evaluation.

Heuristics Miner was the only algorithm with quality values distinct from 0 in empirical evaluation of event log 4. This result only matches with the class obtained by three classifiers, because other four proposed Alpha Miner and the remaining propose Inductive Miner. Discovery algorithms were impacted by the high level of noise distribution in this event log ($nd_4 = 100$). Models obtained from this kind of logs are incomprehensible and have very low values on quality metrics. This situation could be the explanation for mismatching of classifiers with event log 4. Nevertheless, further experimentation with highly noisy event logs is required to prove this hypothesis.

5 Related Work

Evaluation frameworks allow end users to compare the performance of discovery algorithms through empirical evaluation [13]. But, using this framework as a recommendation mechanism is not suitable due to the cost involved on empirical assessments of discovery algorithms. Following the *model-log* method, another evaluation framework, that includes a parameter optimization step, has been proposed [4]. Nevertheless, the negative examples generation created serious performance problems in the experiments with complex event logs [4].

Other proposed solution is based on selecting reference models of high quality and building from these a regression model to estimate the similarity of other process models [5]. Created serious performance problems However, this approach needs reference models for the evaluation and prediction, a requirement that severely limits its application. In multiple real-world environments, where discovery algorithms need to be applied, the process models are not described or are inconsistent and/or incomplete. Besides, this solution assumes that the actual execution of the processes keeps a close relationship with their reference models. But, inexact results can be expected in contexts where features of the actual logs differ from logs artificially generated by the reference models. Furthermore, the construction of a regression model from process model features discards issues such as noise and lack of information on event logs. These issues have a significant impact on the performance of discovery algorithms.

6 Conclusions and Future Work

Event logs features and process characteristics affect the performance of process discovery algorithms. Classical approaches that select discovery algorithms based on empirical assessments are computationally expensive and time consuming.

This paper evaluates the recommendation of discovery algorithms as a classification problem. For this purpose, a knowledge base, with artificially generated cases was built. Cases combine features of event logs and process characteristics with impact on performance of discovery algorithms. Besides, each case contains the values of one quality metric from each quality criterion.

Two ProM plug-ins developed allow to train seven well known classifiers over the knowledge base built. Recommendation of these classifiers match entirely, on

four from five event logs, with the discovery algorithm with best quality values on empirical evaluation. In all cases recommendation through classification was obtained in a significant lower time than through empirical evaluation.

Experimentation with highly noisy logs and multiple classifier systems is suggested as future work. Besides, research is required to apply the proposed approach on event logs from real environments. In this context, low level patterns such as indirect successions and repeated events could be used to extract process characteristics from real event logs.

References

1. van der Aalst, W.M.P.: Process Mining. Discovery, Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011)
2. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A multi- dimensional quality assessment of state-of-the- art process discovery algorithms using real- life event logs. Inf. Syst. **37**, 654–676 (2012)
3. Desai, N., Bhamidipaty, A., Sharma, B., Varshneya, V.K., Vasa, M., Nagar, S.: Process trace identification from unstructured execution logs. In: 2010 IEEE International Conference on Services Computing (SCC), pp. 17–24 (2010)
4. Ma, L.: How to evaluate the performance of process discovery algorithms. Master thesis, Eindhoven University of Technology, Netherlands (2012)
5. Wang, J., Wong, R.K., Ding, J., Guo, Q., Wen, L.: Efficient selection of process mining algorithms. IEEE Trans. Serv. Comput. **99**(1), 1–1 (2012)
6. vanden Broucke, S.K.L.M., Delvaux, C., Freitas, J., Rogova, T., Vanthienen, J., Baesens, B.: Uncovering the Relationship Between Event Log Characteristics and Process Discovery Techniques. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013 Workshops. LNBIP, vol. 171, pp. 41–53. Springer, Heidelberg (2014)
7. Pérez-Alfonso, D., Yzquierdo-Herrera, R., Lazo-Cortés, M.: Recommendation of process discovery algorithms: a classification problem. Res. Comput. Sci. **61**, 33–42 (2013)
8. van der Aalst, W.M.P., Rubin, V., Verbeek, H., Van Dongen, B., Kindler, E., Günther, C.: Process mining: a two-step approach to balance between underfitting and overfitting. Softw. Syst. Model. **9**(1), 87–111 (2010)
9. Rozinat, A., van der Aalst, W.M.P.: Conformance testing: measuring the fit and appropriateness of event logs and process models. In: Bussler, C.J., Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 163–176. Springer, Heidelberg (2006)
10. Ly, L.T., Indiono, C., Mangler, J., Rinderle-Ma, S.: Data transformation and semantic log purging for process mining. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 238–253. Springer, Heidelberg (2012)
11. van Dongen, B.F., Alves de Medeiros, A.K., Wen, L.: Process mining: overview and outlook of petri net discovery algorithms. In: Jensen, K., van der Aalst, W.M.P. (eds.) Transactions on Petri Nets and Other Models of Concurrency II. LNCS, vol. 5460, pp. 225–242. Springer, Heidelberg (2009)
12. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A critical evaluation study of model-log metrics in process discovery. In: Muehlen, M., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 158–169. Springer, Heidelberg (2011)

13. Rozinat, A., Medeiros, A.K.A.d., Günther, C.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: Towards an evaluation framework for process mining algorithms. BPM Center Report (2007)
14. Michie, D., Spiegelhalter, D.J., Taylor, C.C., Campbell, J. (eds.): Machine Learning, Neural and Statistical Classification. Ellis Horwood, Upper Saddle River (1994)
15. Burattin, A., Sperduti, A.: PLG: a framework for the generation of business process models and their execution logs. In: Muehlen, M., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 214–219. Springer, Heidelberg (2011)
16. Weijters, A., van der Aalst, W.M.P., de Medeiros, A.K.A.: Process mining with the heuristics miner-algorithm. Technische Universiteit Eindhoven. Technical report WP 166 (2006)
17. van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A.: Process discovery using integer linear programming. In: van Hee, K.M., Valk, R. (eds.) PETRI NETS 2008. LNCS, vol. 5062, pp. 368–387. Springer, Heidelberg (2008)
18. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 311–329. Springer, Heidelberg (2013)
19. De Medeiros, A., Weijters, A., van der Aalst, W.M.P.: Genetic process mining: an experimental evaluation. Data Min. Knowl. Discov. **14**(2), 245–304 (2007)
20. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
21. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: XES, XESame, and ProM 6. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBIP, vol. 72, pp. 60–75. Springer, Heidelberg (2011)
22. Rozinat, A., Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
23. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Alignment Based Precision Checking. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops 2012. LNBIP, vol. 132, pp. 137–149. Springer, Heidelberg (2013)
24. vanden Broucke, S.K.L.M., De Weerdt, J., Baesens, B., Vanthienen, J.: Improved artificial negative event generation to enhance process event logs. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 254–269. Springer, Heidelberg (2012)
25. De Weerdt, J., Baesens, B., Vanthienen, J.: A comprehensive benchmarking framework (CoBeFra) for conformance analysis between procedural process models and event logs in ProM. In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2013), part of the IEEE Symposium Series in Computational Intelligence 2013 (2013)
26. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009). 07535