Identifying Potential Failures in Business Processes Using Process Mining Techniques

Guillermo Calderón and Marcos Sepúlveda

Computer Science Department
Pontificia Universidad Católica de Chile
gcalder1@uc.cl, marcos@ing.puc.cl

Abstract. One of the most common and costly problems that organizations are facing is to find the source of failures in business processes. These failures are often associated with a missing or unnecessary execution of some process activities; or with how some activities are performed.
We have proposed an automatic technique to identify sources of failures in business process logs. This technique has two stages; first, we look for three different types of potential failures; second, the most likely sources of failures are identified. The first stage is addressed in this article.
Initially, the original event log is filtered on two new logs, the former with cases that produce acceptable results and the latter with cases that produce failed results. Then, *Process Mining* algorithms, specifically *process discovery* and *process conformance*, are combined for identifying potential failures.
We tested this technique by creating different kind of logs. Results show the algorithm is able to successfully identify the potential failures.
Keywords: Business process improvement, process mining.

1 Introduction

Organizations are currently carrying out very comprehensive initiatives to improve their business processes, since they acknowledge process management is fundamental for achieving their business goals in a highly competitive environment. Finding the sources of failures is a common and costly problem in these initiatives.

The result of a business process that has failures is a defective product or service, and becomes visible in several ways, e.g., a customer complaint, a product that does not satisfy quality controls, or the return of a product. These failures are often associated with a missing or unnecessary execution of some process activities; or with how some activities are performed, such as poor performance of business workers, or the usage of incorrect data or inappropriate resources. Currently, there is no automatic technique to help finding these failures.

We have proposed a technique to identify sources of failures in business process logs. This technique has two stages; first, we look for three different types of potential failures; second, the most likely sources of failures are identified. The

first stage is addressed in this article. In the second stage, the technique takes the candidate failures that were initially detected, and then applies adapted *Data Mining* methods to identify the source of failures, considering specific characteristic of business processes, such as, temporal restrictions, sequencing and parallelism. This technique can be used as a starting point for process improvement initiatives.

We have identified three generic types of potential failures: those caused by failing to perform some process activities, those due to the unnecessary execution of some activities, and those cause by other reasons. Among these other reasons are: wrong activities sequencing, unusual temporal execution, or problems found in the execution of single activities.

The first stage aims at finding potential failures of these three generic types. Initially, the original event log is filtered on two new logs, the former with cases that produce acceptable results and the latter with cases that produce failed results. Then, the proposal relies on *Process Mining* techniques, specifically on *process discovery* (to find ideal models) and *process conformance* (to identify failures).

The rest of this article is structured as follows. Section 2 presents the related work to this topic. Section 3 describes the proposed technique, and sections 4 and 5 present the results and conclusions of this work, respectively.

2 Related Work

Process Mining allows discovering and analyzing business processes from event logs obtained from *Process-Aware Information Systems* (PAIS), among them it is possible to highlight *Business Process Management Systems* (BPMS) and corporate systems, such as *Enterprise Resource Planning* (ERP), *Customer Relationship Management* (CRM) and *Supply Chain Management* (SCM). Through this analysis, it is possible to identify certain characteristics that describe their functionality (e.g., bottlenecks, behavioral patterns or business rules). For better detail on the subject, you can review the work of van der Aalst [1] [2].

Of the three main levels of analysis [3], we have considered the discovery and conformance levels (Fig. 1). Specifically, for the discovery stage we have used the *alpha* algorithm [4], due to its simplicity and also that allows a quick and easy integration with our technique, obtaining a business process model in the Petri net format, which is necessary for the next stage.

For the conformance stage, we use the *Conformance Checker* algorithm [5] [6] [7]. This algorithm requires an initial mapping (i.e., an association between the tasks in the process model and those in the event log). We will use the mapping for detecting activities that are not found in the log, but are part of the model. Then, the *Conformance Checker* algorithm uses two mechanisms for comparing the cases in an event log with a given model: *fitness* and *appropriateness*. In our technique, we use only the *fitness* analysis[1]. This method has two indicators:

[1] Level of behavior compliance between the process model and the log, i.e., whether the log fits the model.

Remaining Tasks, which identifies those activities that were not executed in some cases, and *Failed Tasks*, which identifies those activities that were executed when they were not allowed.

3 Proposal

Our approach for finding potential business process failures combines two well-known *Process Mining* techniques: *Process Discovery* and *Process Conformance*.

Our technique begins with the partition of the original event log on two new logs (Fig. 1), the former with cases that produce acceptable results and the latter with cases that produce failed results. This filtering is based on business metrics; therefore, a requirement for using this technique is to have a characterization of the process output for each process instance in the event log, e.g., a business metric that allows distinguishing between successful and failed results. With these two new logs, two process models are created: one that represents the successful cases (how the process should be done) and another one that represents the failed cases (how the process was executed when the outcome was incorrect). Then, two conformance analyses are performed. In the first conformance analysis, the failed-cases log is compared with the model created with the successful-cases log. In the second conformance analysis, the successful-cases log is compared with the model created with the failed-cases log.

Our approach is based on two main assumptions:

- We have a known process; i.e., the business analyst is able to validate the generated model.
- There are reports of the results of the process, i.e., it is known whether the results were acceptable or not, according to some business metrics.

The three steps of our technique are described below.

3.1 Filter

The original event log must have information to distinguish which cases produce successful or failed results. Based on this information, the original log is filtered on two new logs, the former with cases that produce acceptable results and the latter with cases that produce failed results. Generally, this information is associated with a metric of the process, e.g., in an Internet sales process, a metric could be the delivery time; this metric must be stored in the event log as an additional attribute. All those process instances that have not complied with the delivery time will be separated in the failed-cases log, and the remaining ones in the successful-cases log.

We have developed a plug-in in *ProM*, called *Metrics Filter*. It splits the log based on any process attribute in order to get the successful-cases log and the failed-cases log.

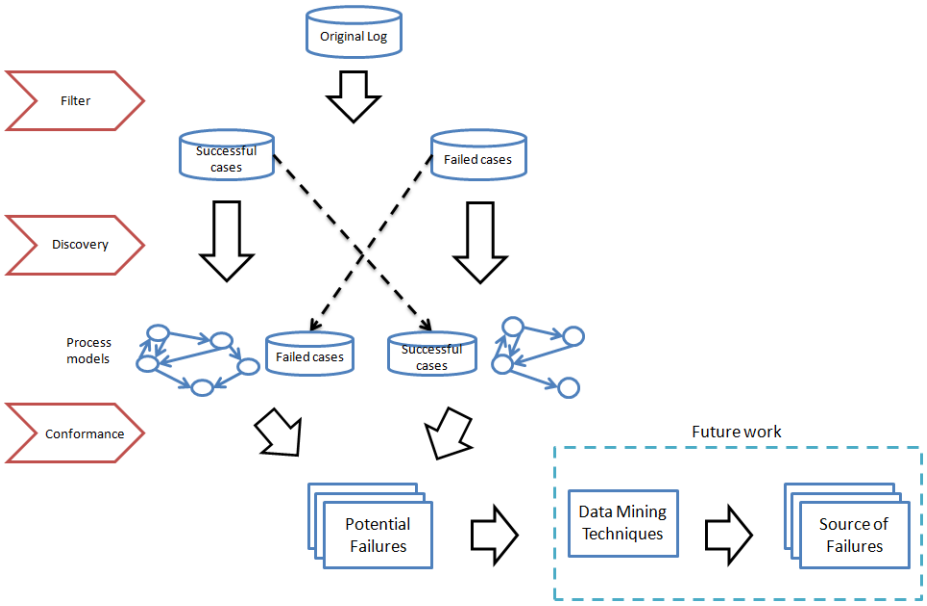

Fig. 1. *Potential Failures detection in business processes. In addition, the future work is shown, where Data Mining techniques will be adapted for identifying the actual source of failures.*

3.2 Discovery

As a second step, a process discovery algorithm is applied to both logs (successful-cases and failed-cases logs), in order to obtain an "ideal" model for each of them. We use the *alpha* algorithm [4] for this task due to its simplicity and also because it allows obtaining a Petri net directly, which will be later on used on the conformance checking step.

We have developed another plug-in in *ProM*, called *Discovering Failures*. In this plug-in, we are planning to integrate the whole technique, but so far, the plug-in only comprises up to the discovery step.

3.3 Conformance

The last step of this technique consists on applying the *conformance checker* algorithm to identify the failures by comparing both logs against the discovered models.

Two conformance analyses are required because the initial *mapping* only works in one direction: the activities in the model are mapped to the activities in the log, but not in the other direction [5]. Since we need mapping the activities in both directions (to find both missing and unnecessary activities), both analyses are needed.

In the first conformance analysis, the failed-cases log is compared with the model created with the successful-cases log. The initial *mapping* and the *Remaining Tasks* indicator of the *fitness* analysis are used for identifying activities that were not executed in the failed cases. These activities represent potential failures because the wrong outcome might have been caused by failing to perform these activities.

In the second conformance analysis, the successful-cases log is compared with the model created with the failed-cases log. As in the previous case, the *mapping* and the *fitness* analysis are used for identifying activities that were not executed in the successful cases. These activities represent potential failures because the wrong outcome might have been caused by the unnecessary execution of these activities.

We use the *conformance checker* algorithm only for detecting which activities are not being executed; therefore we do not need to use all the mechanisms and indicators provided in this algorithm.

The result of this step is a set of activities that were performed differently. These activities can be classified into at least three types of failures.

3.4 Potential Failures Classification

By comparing the two logs, three generic types of potential failures can be identified (Table 1): those caused by failing to perform some activities (Type I), those due to the execution of unnecessary activities (Type II), and those due to other reasons (Type III). Among these other reasons are: wrong activities sequencing, unusual temporal execution, or problems found in the execution of single activities. Even though in this article the three types of failures are identified, the first two types are analyzed more deeply.

Table 1. Types of potential failures

Type	Description
I	Activities are missing in the cases of the failed-cases log
II	There are some unnecessary activities in the cases of the failed-cases log
III	The same activities are found in the cases of both logs

These potential failures might have sub-types (Table 2). The type I failure has three variations:

1. All missing activities do not appear in all cases of the failed-cases log. This means that all missing activities were never performed in all cases that produced failed results.
2. All missing activities appear at least once in the failed-cases log. This means that all missing activities were performed at least in one of the cases that produced failed results.

3. Some missing activities do not appear in the whole failed-cases log; the others appear at least once in the failed-cases log. This means that a subset of missing activities was never performed in all cases that produced failed results. The remaining activities are executed at least once.

The type II failure has also three variations:

1. All unnecessary activities do not appear in all cases of the successful-cases log. This means that all unnecessary activities were never performed in all cases that produced successful results.
2. All unnecessary activities appear at least once in the successful-cases log. This means that all unnecessary activities were performed at least in one of the cases that produced successful results.
3. Some unnecessary activities do not appear in the whole successful-cases log; the others appear at least once in the successful-cases log. This means that a subset of unnecessary activities was never performed in all cases that produced successful results. The remaining activities are executed at least once.

Table 2. Sub-types of potential failures

Sub type	Description
I.1	All missing activities do not appear in the whole failed-cases log.
I.2	All missing activities appear at least once in the failed-cases log.
I.3	Some missing activities do not appear in the whole failed-cases log; the remaining ones are executed at least once.
II.1	All unnecessary activities do not appear in the whole successful-cases log.
II.2	All unnecessary activities appear at least once in the successful-cases log.
II.3	Some unnecessary activities do not appear in the whole successful-cases log; the remaining ones are executed at least once.

3.5 Failure identification

To identify the potential failures, both the initial *mapping* and the *fitness* analysis of the *conformance checker* algorithm are used (Table 3). The following description outlines the mechanisms used for identifying all different failure subtypes.

– Failure I.1 is identified using both algorithms. Since all missing activities are never executed, the *mapping* algorithm finds the inconsistencies between the ideal model and the failed-cases log. In this case, the *Remaining Tasks* indicator of the *fitness* analysis should only highlight the same activities found by the *mapping*.

– Failure I.2 can only be identified with the *fitness* analysis. Since different activities are executed at least in one of the cases, they are mapped correctly, but the *fitness* analysis is able to detect them. The *Remaining Tasks* indicator highlights the missing activities.
– Failure I.3 is identified using both algorithms. The *mapping* identifies the set of activities that are never executed (inconsistency between the model and the log), while the *Remaining Tasks* indicator highlights the missing activities that were executed at least once.
– Failures II.1, II.2 and II.3 are identified in a similar way to the failures I.1, I.2 and I.3, respectively, but using the *conformance checker* algorithms with the model created with the failed-cases log (that represents how the failed cases are performed) and the successful-cases log. By inverting the model and log, we look for unnecessary activities instead of missing activities.
– Failure III is identified when neither the initial *mapping* nor the *fitness* analysis detect any problems in both *conformance checking*. In other words, when neither failures of type I nor type II are found.

Table 3. Detection mechanisms for failure sub-types

Failure	Detection mechanisms for failure sub-types
I.1, I.3, II.1, II.3	*Mapping* and *fitness* analysis
I.2, II,2	*Fitness* analysis
III	When neither *mapping* nor *fitness* analysis detect problems

It might be thought that it would be sufficient to compare the activities and cases of both logs without using the *conformance checker* algorithm, but this procedure would only identify failures I.1 and II.1. Alternatively, we could have created a new algorithm to compare the logs directly to identify potential failure activities of all types. However, our approach takes advantage of the robustness of the *conformance checker* algorithm.

4 Results

We have conducted a semi-automatic evaluation of the proposed technique by performing the activities described below.

Process simulation

We have simulated a real mortgage lending process, which begins with the customer registration. The bank assesses the customer needs and simulates the credit, so as the customer can evaluate the loan conditions. If the client accepts the loan proposition, a risk analysis of the financial conditions of the customer is performed; in this stage, the bank could cancel the application. If the risk

analysis is positive, the bank's legal department inquires about the records of the real property to evaluate them and makes an appraisal. If discrepancies are found in the records, the application could be cancelled. Once this last stage is passed, the contract is elaborated and the bank emits the payment. The client signs the papers and the money is delivered.

We have used the *CPN Tools*[2] for this simulation, which allows modeling and simulating any kind of process. By adding a set of pre-built functions to the *CPN Tools* [8], it is possible to generate event logs in a XML format. Then, using the *ProMImport*[3] tool, all generated logs are grouped in a MXML file.

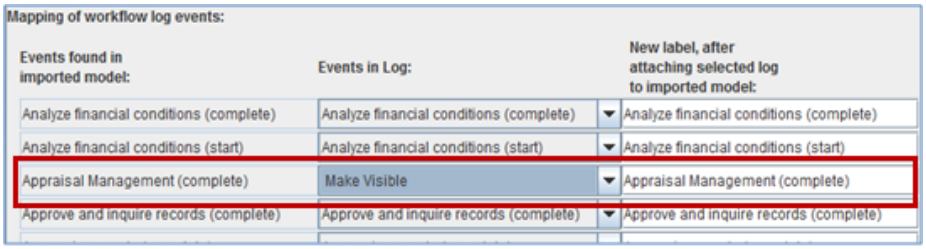

Fig. 2. *Detecting failure I.1 in ProM with the mapping algorithm. The Appraisal management activity is never executed in the failed-cases log; therefore the mapping algorithm finds an inconsistency between the model and the log. In order to verify no other activity has been executed in some cases, a fitness analysis must be performed; for this purpose, this activity must be made visible.*

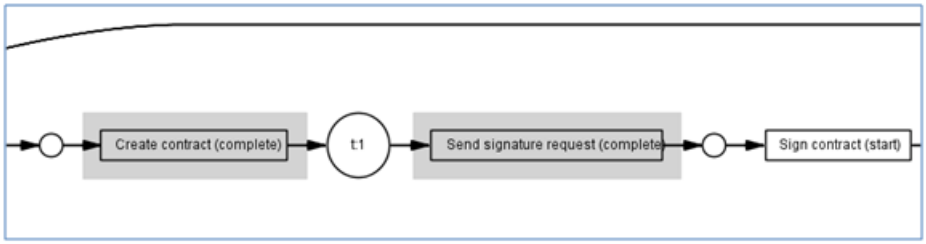

Fig. 3. *Detecting failure I.2 in ProM. The Create Contract and Send signature request activities are executed in some of the failed cases, therefore the mapping algorithm does not find any inconsistency, but the Remaining Tasks indicator of the fitness analysis allows identifying them.*

[2] http://wiki.daimi.au.dk/cpntools/cpntools.wiki
[3] http://prom.win.tue.nl/research/wiki/promimport/start

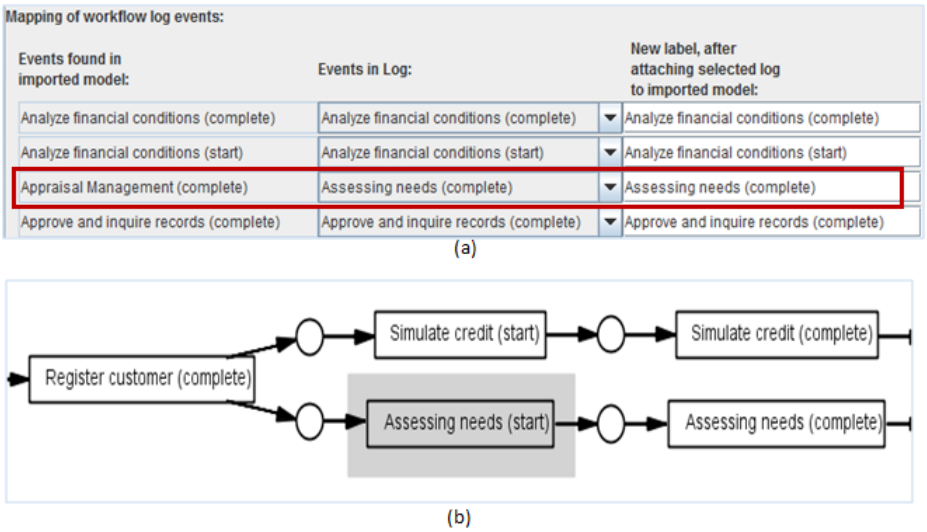

Fig. 4. *Detecting failure I.3 in ProM. (a) The mapping algorithm allows identifying that the Appraisal management activity is never executed in the failed-cases log. (b) In the same log, the Assessing needs activity is executed in some of the failed cases; therefore the mapping algorithm does not find this inconsistency, but the fitness analysis is able to detect it.*

Failed cases creation

We have created a total of 16 event logs; in each of them we have introduced different failure sub-types. A sensitivity analysis of the technique was considered, i.e., we have created logs varying the percentage of failures in each of them. For example, for I.2 failures, we have created logs with 1, 5, 20, 50 70 and 90 percent of failed cases.

For failures I and II and its variations, different activities have been removed, e.g., the Appraisal management activity in one log (Fig. 2) and the Create contract and Send signature request in another one (Fig. 3). For failure III, the original process was simulated without modifications. All these simulations are based on real cases, representing actual organizational situations.

Discovery

Two models are created for describing the successful-cases and failed-cases logs. We have used the following algorithms implemented in *ProM* to model them:

- *Alpha* algorithm, to model the mortgage lending process.
- *Petri Net Kernel File*, to convert the model into a Petri Net format, which is required for the conformance stage.

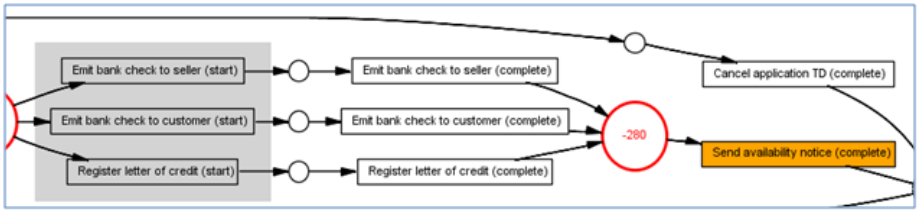

Fig. 5. *Detection of a special case: failure in Alternative Paths. In this case, only one activity has not been executed (either Emit check to seller, Emit bank check to customer, or Register letter of credit). This problem can be detected using the Failed Tasks indicator. By having a single Task Failed, we can deduce that only a single activity was not performed. In this case, the technique singles out the three activities as potential failure activities.*

Conformance

For the Conformance step, the successful-cases and the failed-cases logs have been evaluated separately. We have used the *Conformance Checker* algorithm (implemented in *ProM*) to identify the failures.

Some results can be seen in Figs. 2 to 7. Figs. 2 to 4 show the identification of different sub-types of failure I.

Special situations have also been considered. Fig. 5 shows a situation where are alternative paths; in this case, the *Failed Tasks* indicator of the *fitness* analysis is used to detect that only one of the alternative activities was not executed. In a similar way, Fig. 6 and Fig. 7 show situations with parallelism and activities in the wrong order, respectively.

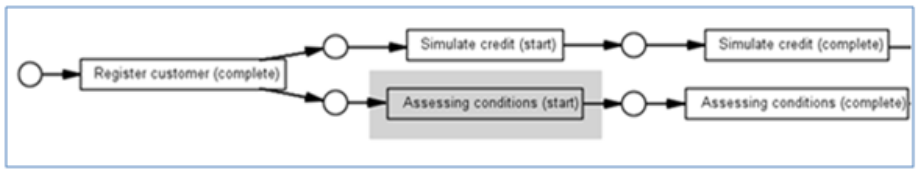

Fig. 6. *Detection of a special case: Parallel Paths. If any parallel activity is not executed, it can also be detected. The activity Assessing Conditions is detected despite being in a parallel path.*

Evaluation

It has been possible to identify 100% of the failures, of all types and sub-types (Tables 1 and 2), following the semi-automatic evaluation. Since this identification is binary, i.e., the failure is identified or not, and the identification process

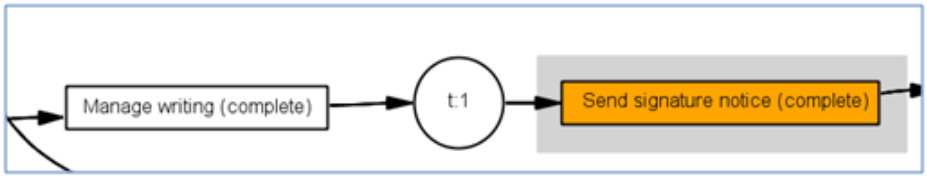

Fig. 7. *Detection of a special case: activities in the wrong order. The activities Manage writing and Send Signature notice have been executed in the opposite order in some failed cases. It can be identified using both the Failed Tasks and the Remaining Tasks indicators of the fitness analysis. In this case, the Send signature notice is highlighted to be a Remaining task and a Failed task at the same time. In this case, the technique single out both activities (Send signature notice and Manage writing) as potential failure activities. Data mining techniques will have to find later on the inconsistency in the activities sequencing.*

is deterministic, the level of certainty of the evaluation is absolute. Moreover, there is no restriction on the nature of the processes for the proposed technique.

Different logs were created to evaluate the sensitivity of the algorithm. This assessment also showed positive results. Regardless of the percentage of failed cases, the results do not suffer any distortion.

As was explained at the beginning of this article, the global technique that allows finding sources of failures in a business process has two stages: (i) to identify different type of potential failures and (ii) to identify the sources of failures. We have considered simulated cases to verify the approach introduced in this paper for the first stage. The second stage and the whole technique will be validated using real event logs provided by one of the largest telecommunication companies in Chile.

5 Conclusions and future work

We have proposed an automatic technique to identify potential business process failures based on event logs. This technique is able to identify seven varieties of failures, classified into three generic types. We tested this technique by creating different kind of logs. Results show the algorithm is able to successfully find missing or unnecessary activities that might explain a wrong outcome.

This technique will enable organizations to reduce time and costs in carrying out business process improvement initiatives; and hopefully, to boost their performance.

Currently, we are working on the full automation of this technique (as a plug-in for *ProM*) and on the formalization of the different types and subtypes of potential failures. Also as a future work, we are planning to adapt *Data Mining* techniques for identifying the actual sources of failures.

References

1. van der Aalst, W.M.P., Weijters, A.J.M.M.: Process Mining: a research agenda. Computers in Industry 53(3), 231–244 (2004)
2. van der Aalst, W.M.P., Dongen, B.F.V., Herbst, J., Maruster, L., Schimm, G., Weijters A.J.M.M.: Workflow mining: A survey of issues and approaches. Data and Knowledge Engineering 47(2), 237–267 (2003)
3. van der Aalst, W.M.P.: Process mining and monitoring processes and services: Workshop report. The Role of Business Processes in Service Oriented Architectures 6291, (2006)
4. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128-–1142 (2004)
5. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Information Systems 33(1), 64–95 (2008)
6. Rozinat, A., van der Aalst, W.M.P.: Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models. Business Process Management Workshops 3812, 163-–176 (2006)
7. Rozinat, A., van der Aalst, W.M.P.: Conformance testing: measuring the alignment between event logs and process models. BETA Working Paper Series, WP 144, Eindhoven University of Technology, Eindhoven, (2005)
8. de Medeiros, A.K.A., Günther, C.W.: Process Mining: Using CPN Tools to Create Test Logs for Mining Algorithms. In: Proceedings of the Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, 177-190 (2005)