# Automatic discovery of failures in business processes using Process Mining techniques

**Guillermo Calderón-Ruiz[1], Marcos Sepúlveda[2]**

[1]Departamento de Ciencias e Ingenierías Físicas y Formales – Universidad Católica de Santa María (UCSM) – Arequipa – Perú

[2]Departamento de Ciencia de la Computación – Pontificia Universidad Católica de Chile Santiago – Chile

gcaldero@ucsm.edu.pe, marcos@ing.puc.cl

***Abstract****—One of the most common and costly problems that organizations are facing is to find the causes of failures in business processes. Failures are often due to missing or unnecessary execution of some process activities; or with how some activities are performed. Currently, there is no automatic technique that helps finding these causes. We propose a novel technique to identify potential causes of failures in business process by extending available Process Mining techniques. Initially, the original event log is filtered in two logs, the former with successful cases and the latter with failed cases. Then, behavioral patterns are extracted from both event logs using the Performance Sequence Diagram Analysis algorithm. Finally, both sets of patterns are compared considering control flow and time perspectives. The differences found represent potential causes of failures in business processes. We tested this technique using several synthetic event logs. Results show the technique is able to successfully find missing or unnecessary activities, and failed behavioral patterns that differ from successful patterns either in the control flow or in the time perspective.*

## 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 causes of failures in a business process has become one of the most common and costly problems in these initiatives.

A failure is identified when the execution of a business process generates a result (product or service) that does not fulfill the target expectations. 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 due to missing or unnecessary execution of some process activities; with how some activities are performed, such as the execution flow or the processing time; or with the data used to perform some activities.

Finding the source of failures (root cause analysis) is a process based on semi-formal techniques and brainstorming, Andersen & Fagerhaug (2006), but these techniques are difficult to systematize and resource-consuming. These techniques can be classified into two, Hongtao, Yong, & Jiansa (2006): participatory and analytical techniques.

Participatory techniques organize ad-hoc multidisciplinary teams in order to solve a problem through workshops and other collaborative activities. According to experts, these techniques require tedious and very subjective processes, Sharp & McDermott (2008).

The analytical techniques use principles or formal theories created specifically for this type of problem, e.g., Michael Hammer's Process and Enterprise Maturity Model, Hammer (2007).

Currently, there is no automatic technique to support and expedite the search for causes of failures in business processes. In this document, we propose a novel technique to identify potential causes of failures in business process by extending Process Mining algorithms.

*Process Mining* allows discovering and analyzing business processes based on event logs. Through this analysis, certain characteristics that describe process functionality (e.g., bottlenecks, behavioral patterns, or business rules) can be identified. For further detail on the subject, you can review Van der Aalst, Weijters, & Maruster (2004), Wil van der Aalst (2004), Van der Aalst (2011).

Our proposal is divided into three stages (Figure 1). First, in the *Filter* stage, the original event log is filtered in two logs, the former with successful cases and the latter with failed cases. Then, in the *Pattern Discovery* stage, behavioral (or execution) patterns are extracted from both event logs, i.e., extracting patterns that produce acceptable results and patterns that produce failed results. The *Performance Sequence Diagram Analysis* algorithm, Hornix (2007) is used to extract these patterns. We selected this algorithm because it considers control flow (the order in which activities are performed) and time (tasks' duration and waiting times between tasks) features.

Finally, we compare the failed patterns against the patterns that produce acceptable results; we have called this stage *Behavioral Pattern Conformance*. This comparison takes into account control flow and time features, e.g., sequencing of tasks or processing times. The differences found represent potential causes of failures in the analyzed business process.

We have identified four types of potential failures: *missing tasks* (when some tasks are not performed), *unnecessary tasks* (unnecessary execution of some tasks), *different behavior* (execution patterns different from normal sequencing), and *different timing* (durations different to normal executions).

The rest of this document is structured as follows. Section 2 presents the related work and the comparison with other techniques. Our proposed technique is detailed in

Section 3. Section 4 describes an evaluation of the technique. Finally, the conclusions are presented in Section 5.

## 2. Related Work

We have found three kinds of works that could be considered related to our proposal. These reports did not directly address the problem of finding causes of failures in business processes in an automatic way.

Heravizadeh, Mendling, & Rosemann (2008) proposes a conceptual methodology of root-cause analysis in business processes, based on the definition of softgoals (non-functional requirements) for all process activities, as well as correlations between these softgoals and related quality metrics. This methodology has two important differences with our proposal; first of all, requires much effort from participants to document requirements, relationships and respective metrics, and second, it is not automatic.

Bezerra & Wainer (2008) and Van der Aalst & De Medeiros (2005) present mechanisms to identify anomalies in business processes executions, by using available *Process Mining* techniques in *ProM*[1] (e.g., the *alpha* algorithm plug-in or the *control-flow benchmark* plug-in). These mechanisms are not designed to find the source of failures, but to find strange executions within a process. They also require much interaction with the user to get a solution.

Bose & Suresh (2008) presents two mechanisms to identify the root cause of failures in software programs. For this purpose, Bose uses the sequence of events that led to the failure in order to identify common patterns of execution. This is an important difference to our proposal, since Bose uses only the sequence of events until the failure occurs, i.e., the execution stops when the failure appears, so it is known where the failure is. In event logs, when the process is executed until the end, we do not know where the failure is.

Besides, we use the *Performance Sequence Diagram Analysis* algorithm, Hornix (2007), which is a means to assess the performance of business processes. Specifically, it allows identifying behavioral patterns, i.e., executions that have the same sequencing behavior. Therefore, it is possible to identify certain characteristics of the processes, e.g., most common execution patterns or patterns that exceed the normal execution time. This algorithm shows these patterns as sequence diagrams (Figure 2), where the blocks are related to activities and the arrows are related to waiting times between activities. Currently, many studies use patterns to analyze and improve business processes, Souza, Azevedo, & Santoro (2012), our work is embedded within this category of works.

As we said, there are not automatic techniques that allow finding the source of failures in a business process. Despite this, we found two *Process Mining* techniques that could be used as a starting point to solve this identification problem.

The first of these techniques is the *Conformance Checker* algorithm, Rozinat & Van der Aalst (2008). This algorithm allows identifying failed activities in a process, i.e.,

[1] ProM is a pluggable and open-source framework for Process Mining

activities that are not executed. But this algorithm does not analyze this problem; neither assures these failed activities are related to wrong executions in the process.

The second of these techniques is the Anomaly detection algorithm, Bezerra & Wainer (2008). This algorithm classifies the cases in an event log into normal and anomalous, taking into account the control flow perspective. As the Conformance Checker, this algorithm does not analyze the anomalous cases; neither assures these anomalous cases are related to wrong executions in the process.

Our proposal has the following main differences:

- It identifies failures in the control flow and time perspective.
- It identifies the potential sources of these failures.
- Works in an automatic way.

## 3. Developed Technique

We propose a novel technique to identify potential causes of failures in business process logs by extending available *Process Mining* techniques, specifically the Performance Sequence Diagram Analysis plug-in (a *Process Mining* technique implemented in *ProM*).

The three steps of our proposal are described below and are shown in Figure 1.

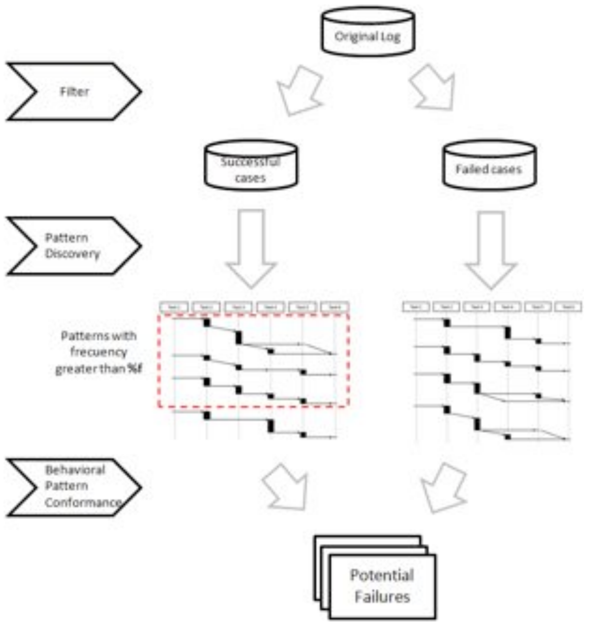


**Figure 1. Discovering potential failures in business processes in three stages**

## 3.1 Filter

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. 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. 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 a successful-cases log and a failed-cases log.

## 3.2 Pattern Discovery

As a second step, behavioral (or execution) patterns are extracted from both event logs, i.e., patterns that produce acceptable results (successful executions) and patterns that produce failed results (failed executions). *The Performance Sequence Diagram Analysis* plug-in is used to perform this step.

Before proceeding with the next step, the successful patterns are filtered considering a minimum frequency of appearance (*%f*). In this way, the successful patterns that have low frequencies are put aside, since they could represent strange or undesirable behavior.

## 3.3 Behavioral Pattern Conformance

In the last step, we compare both sets of patterns (i.e. successful patterns vs. failed patterns). The comparison is performed by taking each failed pattern and comparing it against all successful patterns. The differences found represent the potential causes of failures in the analyzed business process.

It is possible to identify four types of differences (i.e., potential causes of failures):

- *Missing tasks*. Those activities that are not performed in any failed pattern but are performed in all successful patterns.

  Not performing these activities may be a likely cause for getting undesirable results in the business process.

- *Unnecessary tasks*. Those activities that are performed in some failed patterns but not in any successful pattern.

  Performing these activities may be a likely cause for getting undesirable results in the business process.

- *Different behavior*. Those failed patterns that are not equivalent to any successful patterns. These failed patterns may be a likely cause for getting undesirable results in the business process.

- *Different timing*. When a failed pattern has an equivalent pattern among the successful patterns, i.e., both have the same flow, the time perspective is analyzed: processing time, tasks' duration, and waiting times are compared between the equivalent patterns. When the time difference exceeds a value defined by business experts, a potential cause of failure is found.

These four potential causes of failures consider two business process perspectives: the control flow and time perspectives. The first three potential causes of failures (i.e., *missing* and *unnecessary* tasks, and *different behavior*) are from the control flow perspective. The last potential cause of failure (i.e., *different timing*) is from the time perspective.

## 3.4 Identifying failures in the control-flow perspective

To understand how the control-flow causes of failures are identified, we will use the following graphic examples.

- *Missing tasks*. Consider the successful patterns in Figure 2 and the failed patterns in Figure 3. By comparing each failed pattern against all successful patterns, we could identify that Task 5 is never executed in any failed patterns. Not executing an activity could be the reason for wrong outcomes in a business process.
- *Unnecessary tasks*. Comparing the same examples in Figure 2 and Figure 3, we could identify that Task 6 is executed in some failed patterns but never in the successful ones. Executing extra activities could be the reason for wrong outcomes in a business process.

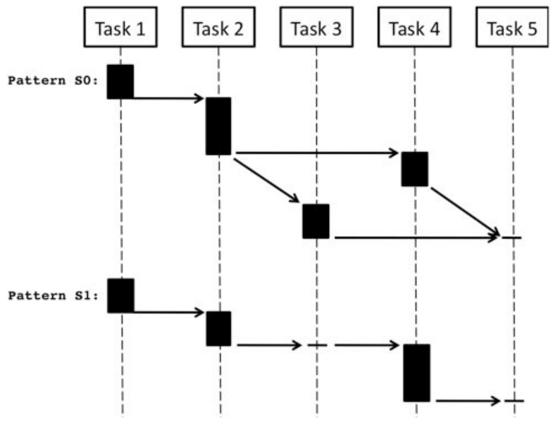


**Figure 2. Examples of successful patterns. These patterns are shown as sequence diagrams.**

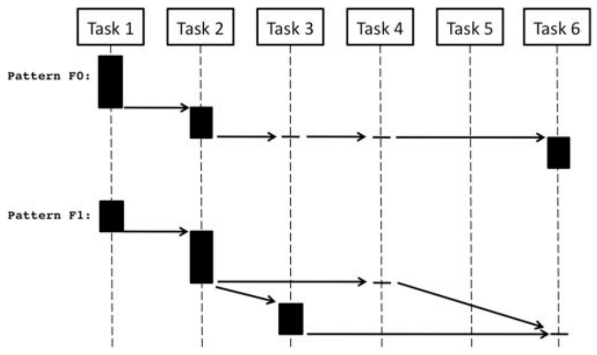


**Figure 3. Examples of failed patterns, with missing and unnecessary tasks.**

- *Different behavior*. For this source of failure, consider the successful patterns in Figure 2 and the failed patterns in Figure 4. By comparing the failed pattern F2 against all successful patterns (S0 and S1), we can observe that pattern F2 has a different sequence behavior. Executing activities in different sequencing could be the reason for wrong outcomes in a business process.

  If we continue the comparison (i.e., pattern F3 against all successful patterns), we can identify that pattern F3 is equivalent to successful pattern S0. In this case, we should evaluate another business process perspective (e.g., time or data) to find out the reason for wrong outcomes in pattern F3.

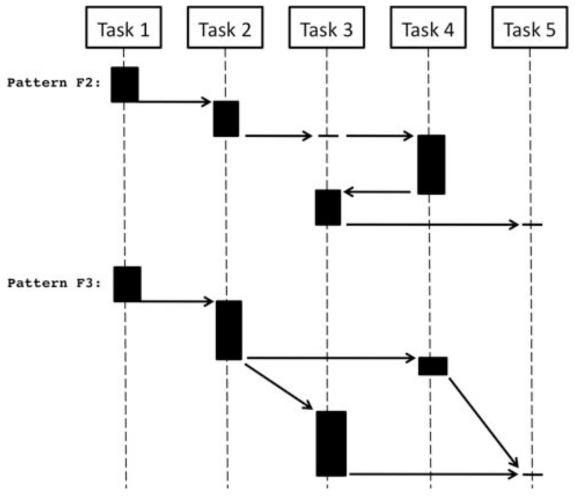


**Figure 4. Examples of failed patterns. Some different behavior and time-related failures are shown.**

## 3.5 Identifying failures in the time perspective

If a failed pattern is equivalent to a successful pattern, it is necessary to make a time-based evaluation. For this, the processing time, the duration of all tasks (blocks) and the waiting times (arrows) are compared with the corresponding time metrics in the equivalent successful pattern. It is necessary to have some parameters to apply this comparison, these are:

- *Processing time* threshold (*%U*) for normal executions. It represents an acceptable percentage variation of the processing time.
- *Duration* threshold ($\%T_i$) for each task *i*.
- *Waiting* threshold ($\%W_j$) for each waiting time *j* between tasks.

A global threshold for all task durations and waiting times could be considered, if there is not detailed information about the process. For example, 10% for the overall processing time, 5% for all task durations and 8% for waiting times could be considered.

In the time-based verification, first the processing time of the failed pattern is compared against the processing time of the successful pattern, considering the processing time threshold *%U*. If they are different, a *Different-timing* potential cause of failure has been identified.

If the processing time of the failed pattern is within the limits of the threshold *%U*, the duration of each task (i.e., duration of the block) is compared. If the duration of a task in a failed pattern is greater or less than the defined threshold $\%T_i$, a *Different-timing* potential cause of failure has been identified.

Also, the waiting times between tasks (i.e., the duration time associated to arrows) are compared. If the result is greater or less than the defined threshold $\%W_j$, a *Different-timing* potential cause of failure has been identified.

To clarify the algorithm, let us show the following graphic examples. For these examples consider the successful patterns S0 and S1 (Figure 3). Also, consider the failed pattern F3 (Figure 4). It can be seen that pattern F3 is equivalent to pattern S0 (they have the same blocks and arrows, and in the same order). The technique identifies two failures:

- The duration of tasks 3 (in pattern F3) is greater than its peer (in Pattern S0) and it exceeds the threshold $\%T_3$. Therefore, a *Different-timing* potential cause of failure is identified
- The duration of tasks 4 (in pattern F3) is lower than its peer (in Pattern S0) and it exceeds the threshold $\%T_4$. Therefore, a *Different-timing* potential cause of failure is identified

To clarify some concepts, let us explain a real business process. The process of *Handling Customer Complaints* begins with a customer call explaining a problem or with an identified problem submitted by the Services area. For each problem, a ticket is created. The ticket should be assigned to a qualified employee for resolution. If the employee cannot solve the problem, it is delegated to another entity (this task can be repeated many times).

Once an employee accepts a ticket, it should be analyzed and solved (which often does not happen). Finally, the ticket is closed.

A successful pattern –in the control-flow perspective- for this process could be:

- Create Ticket – Assign Ticket – Delegate Ticket – Analyze Problem (start) – Try solutions – Analyze Problem (complete) – Close Ticket

A failed pattern –in the control-flow perspective- for this process could be:

- Create Ticket – Assign Ticket – Delegate Ticket – Analyze Problem (start) – Close Ticket

## 4. Evaluation

The technique was implemented in *ProM*[2] and tested with several synthetic event logs. Figure 5 shows the main screen of the implemented plug-in, where it is possible to see the set parameters and some failures in the control-flow and time perspectives. To generate these event logs, we used *CPN Tools*[3] to model and simulate a process. By adding a set of pre-built functions to *CPN Tools,* De Medeiros & Günther (2005), it is possible to generate event logs in a *XML* format. Then, using the *ProMImport*[4] tool, Günther & Van der Aalst (2006), all generated logs are grouped in a *MXML* file[5]. All generated event logs had 1,000 cases, where 844 were successful cases and 156 failed cases.

251 event logs were created and different failure types were introduced in different event logs. In the control-flow perspective, the following failure types were introduced:

- Missing task(s). Those activities that are not performed in any failed pattern, but are performed in all successful patterns.
- Unnecessary task(s). Those activities that are performed in some failed patterns, but are not performed in any successful pattern.
- Omitting task(s) in some failed cases.
- Change in the order of activities in the failed cases.
- Combinations of the above failures.

In the time perspective, the following failure types were introduced:

- Longer or shorter processing times.
- Longer or shorter task durations. In different failed cases, some cases with very long task durations, others with very short task durations. Also, both types of problems in the same case.

[2] Link to plug-in and examples: http://dcc.puc.cl/process-mining/ProM/puc-plugins/.

[3] http://wiki.daimi.au.dk/cpntools/cpntools.wiki

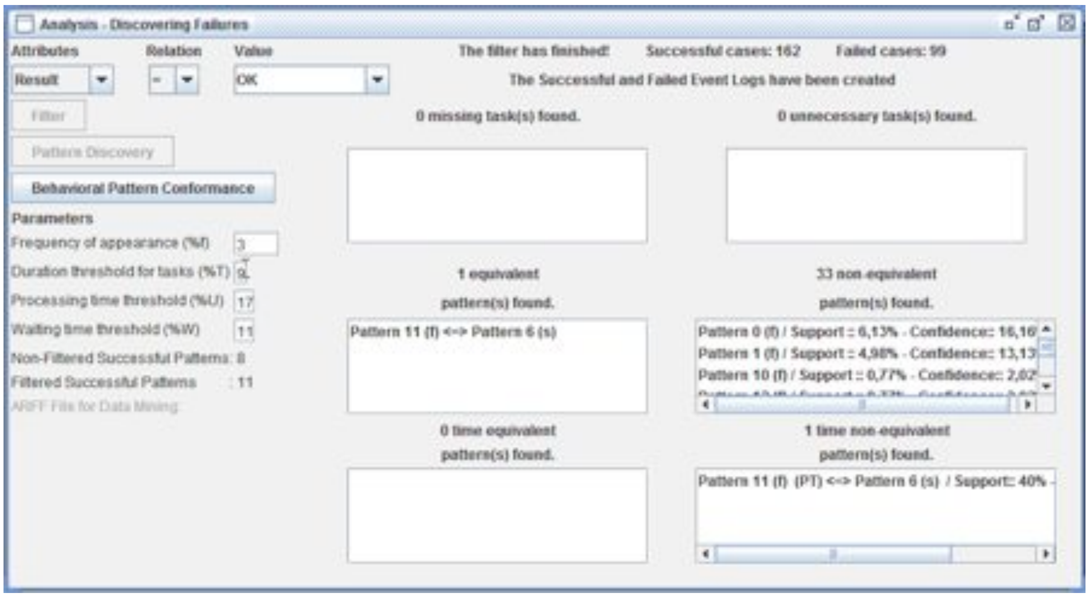
[4] http://prom.win.tue.nl/research/wiki/promimport/start

[5] Special XML format used by the ProM framework.

- Longer or shorter waiting times. In different failed cases, some cases with very long waiting times, others with very short waiting times. Also, both types of problems in the same case.
- Combinations of time failures into different cases. Also, combinations of time failures into a single case.

In addition, sensitivity tests were performed with the following changes:

- Event logs with a very low percentage of failed cases were created (1%, 5%, 10% and 20%).
- Event logs with a very high percentage of failed cases were created (80%, 90%, 95% and 99%).


Analysis - Discovering Failures
Filter
Pattern Discovery
Behavioral Pattern Conformance

Attributes: Result
Relation: =
Value: OK

The filter has finished! Successful cases: 162 Failed cases: 99

The Successful and Failed Event Logs have been created

Parameters

Frequency of appearance (%f) 3
Duration threshold for tasks (%T) 9
Processing time threshold (%U) 17
Waiting time threshold (%W) 11
Non-Filtered Successful Patterns: 8
Filtered Successful Patterns : 11
ARFF File for Data Mining

0 missing task(s) found.

0 unnecessary task(s) found.

1 equivalent pattern(s) found.

Pattern 11 (f) <-> Pattern 6 (s)

33 non-equivalent pattern(s) found.

Pattern 0 (f) / Support :: 6,13% - Confidence:: 16,16
Pattern 1 (f) / Support :: 4,98% - Confidence:: 13,13
Pattern 10 (f) / Support :: 0,77% - Confidence:: 2,02

0 time equivalent pattern(s) found.

1 time non-equivalent pattern(s) found.

Pattern 11 (f) (PT) <-> Pattern 6 (s) / Support:: 40% -

**Figure 5: Main screen of the developed plug-in.**

Results show the technique is able to identify potential causes of failures, i.e., missing or unnecessary activities and behavioral patterns that differ from each other in the control flow or time perspectives.

Our proposal has a global effectiveness of 86%. It has some problems by identifying combined problems in the time perspective, i.e., if one case has more than one failure in the time perspective, the technique only identifies the first one. For example, one case has a very long processing time, activity A has also a very long duration, and activity F has a very short duration. In this situation, our technique only identifies the long processing time.

## 5. Conclusions

We have proposed a novel technique to identify potential causes of failures in business process based on event logs. This technique is able to identify four types of causes. We tested this technique using a simulated process. Results show the technique is able to successfully find missing or unnecessary activities, and behavioral patterns that differ from each other in the control flow or the time perspective. These results might explain wrong outcomes.

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

This technique could be introduced into the emerging methodologies of Process Mining, Van der Aalst (2011), specifically in the Operational Support phase, enabling process managers to reduce time and costs in finding potential causes of failures in a business process; and thus facilitating business process improvement initiatives.

Currently, we are working on improving the technique by adding the comparison in the data perspective. When successful and failed patterns are equivalent in the control-flow and time perspectives, we will analyze the data in the process to try finding the source of failures in this perspective.

## References

Andersen, B., & Fagerhaug, T. (2006). *Root cause analysis: simplified tools and techniques*. ASQ Quality Press.

Bezerra, F., & Wainer, J. (2008). Anomaly detection algorithms in logs of process aware systems. *Proceedings of the 2008 ACM symposium on Applied computing* (pp. 951-952).

Bose, R. P. J., & Suresh, U. (2008). Root Cause Analysis Using Sequence Alignment and Latent Semantic Indexing. *Proceedings of the 19th Australian Conference on Software Engineering*, 367–376.

De Medeiros, A. K. A., & Günther, C. (2005). Process mining: Using cpn tools to create test logs for mining algorithms. *Proceedings of the Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, *576*, 177-190.

Günther, C., & van der Aalst, W. M. P. (2006). A generic import framework for process event logs. *In Eder, J., Dustdar, S., eds.: BPM 2006 Workshops, Proceedings of the Workshop on Business Process Intelligence (BPI2006)*, *4103*, 81–92.

Hammer, M. (2007). The process audit. *Harvard Business Review*, *85*(4), 92-104.

Heravizadeh, M., Mendling, J., & Rosemann, M. (2008). Root Cause Analysis in Business Processes. *QUT ePrints, Queensland University of Technology*.

Hongtao, T., Yong, C., & Jiansa, L. (2006). Architecture of process mining based business process optimization. *IET Conference Publications*, *2006*(CP524), 1066-1069.

Hornix, P. T. (2007). *Performance analysis of business processes through process mining* (Master Thesis). Eindhoven University of Technology, Eindhoven.

Rozinat, A., & van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems*, *33*(1), 64-95.

Sharp, A., & McDermott, P. (2008). *Workflow Modeling: Tools for Process Improvement and Applications Development*. Artech House Publishers.

Souza, A., Azevedo, L. G., & Santoro, F. (2012). Um Método para Selecão de Padrões para Melhoria de Processos de Negócio. *Proceedings of theVIII Simpósio Brasileiro de Sistemas de Informação (SBSI 2012) Trilhas Técnicas*, 279-290.

Van der Aalst, W. M. P. (2011). *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer.

Van der Aalst, W. M. P., & de Medeiros, A. K. A. (2005). Process Mining and Security: Detecting Anomalous Process Executions and Checking Process Conformance. *Electronic Notes in Theoretical Computer Science*, *121*, 3-21.

Van der Aalst, W. M. P., Weijters, A. J. M. M., & Maruster, L. (2004). Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, *16*(9), 1128-1142.

Wil van der Aalst, T. W. (2004). Workflow Mining: Discovering Process Models from Event Logs. http://www.computer.org/portal/web/csdl/doi/10.1109/TKDE.2004.47